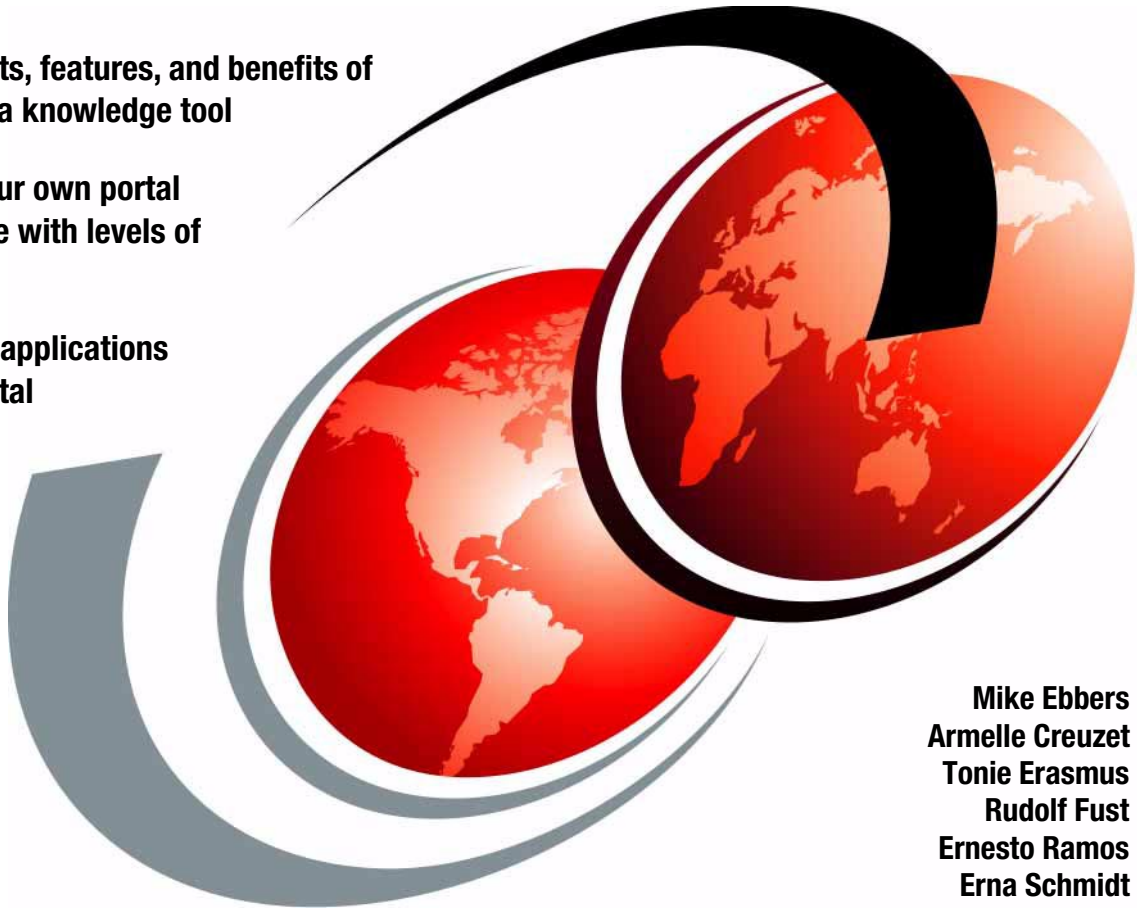
Lotus

IBM

Build a Portal with Domino:

A S/390 Example

- The concepts, features, and benefits of a portal as a knowledge tool
- Building your own portal architecture with levels of content
- Integrating applications into the portal



Mike Ebbers
Armelle Creuzet
Tonie Erasmus
Rudolf Fust
Ernesto Ramos
Erna Schmidt

Redbooks

ibm.com/redbooks

SG24-6231-00

International Technical Support Organization

Build a Portal with Domino: A S/390 Example

April 2001

Take Note!

Before using this information and the product it supports, be sure to read the general information in Appendix C, “Special notices” on page 265.

First Edition (April 2001)

This edition applies to Release 5.06 of Domino for S/390 V5R5, Program Number 5655-B86 for use with the IBM OS/390 Version 2 Release 6 Operating System and above.

Comments may be addressed to:
IBM Corporation, International Technical Support Organization
Dept. HYJ Mail Station P099
2455 South Road
Poughkeepsie, NY 12601-5400

When you send information to IBM, you grant IBM a non-exclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

© Copyright International Business Machines Corporation 2001. All rights reserved.
Note to U.S Government Users – Documentation related to restricted rights – Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract with IBM Corp.

Contents

Figures ix

Preface xiii
The team that wrote this redbook xiii
Comments welcome xv

Part 1. Introduction to S/390 and portal concepts 1

Chapter 1. Portal concepts 3
1.1 Different kinds of portals 4
1.1.1 Horizontal versus vertical portals 4
1.1.2 Corporate portal 5
1.2 Key features of a portal 6
1.3 So what makes a portal a portal 8
1.4 Benefits, elements, and first steps 9
1.4.1 Benefits 9
1.4.2 Elements of knowledge management 9
1.4.3 First steps 11

Chapter 2. Why isn't your Domino server on a S/390? 13
2.1 Domino in a S/390 environment 14
2.2 The value of a large Domino server 15
2.2.1 Less administration effort 16
2.2.2 Access to current data 17
2.2.3 Faster mail routing 17
2.2.4 Less disk space 17
2.2.5 Less network traffic 17
2.2.6 Less processor capacity 18
2.3 Other unique S/390 benefits 18
2.3.1 Multiple applications on one server 18
2.3.2 Running Domino alongside enterprise applications 19
2.3.3 Multiple operating systems on one server 20
2.3.4 Low costs 21
2.3.5 Security and integrity 21
2.3.6 Reliability and availability 21
2.3.7 Growth and flexibility 22
2.3.8 Powerful I/O capability and backup speed 23
2.3.9 Data management 23
2.3.10 Low risk 23
2.4 Developing Notes applications on a S/390 platform 24
2.4.1 Gathering the prerequisites 24

© Copyright IBM Corp. 2001

2.4.2 Creating a support team 25
2.4.3 Service strategies 26
2.4.4 Writing the portal 26

Part 2. Creating the portal architecture 27

Chapter 3. Kernel portal architecture 29
3.1 Individualized levels of content 29
3.1.1 Levels of content 29
3.2 Designing the kernel portal application 31
3.2.1 Portal structure 32
3.3 The portal database 33
3.3.1 Mapping the individualized levels in the application 33
3.3.2 Using access control for content mapping 34
3.3.3 ACL user roles 35
3.4 Designing the main portal page 36
3.4.1 Designing the portal menu structure 37
3.4.2 Designing the enterprise portal 40
3.5 Designing the personal portal pages 41
3.6 Designing the functional portal pages 42
3.7 Designing the administration section 43
3.8 Extranet and Internet applications 43
3.9 Special considerations 44
3.9.1 Security aspects 44
3.9.2 Network traffic 44
3.9.3 The usage of scripts and formulas 45
3.9.4 The layout of the user interface 47
3.9.5 Methods used to open objects 47
3.9.6 Domino for S/390 and Web server integration 48
3.9.7 Domino and WebSphere 49
3.9.8 Methods to open the portal 50

Chapter 4. User personalization 55
4.1 Personalization with user roles 55
4.1.1 How to set up roles 56
4.2 Making a multi-lingual portal 59
4.2.1 Tools for Web site translation 60
4.2.2 Using Domino Global Workbench 60
4.2.3 Database translation process 61
4.2.4 Multilingual database generation 62
4.2.5 Reviewing the glossary 68
4.2.6 Managing design changes 72
4.2.7 Implementing the multilingual databases 73

Chapter 5. Portal security 77
5.1 Authentication and access control 77
5.2 Domino server and HTTP authentication 77
5.2.1 Domino server authentication 77
5.2.2 HTTP server authentication 78
5.2.3 Secure Sockets Layer. 91
5.3 Database access control 98
5.3.1 Access control lists. 98
5.3.2 Roles 99
5.3.3 Groups 99
5.3.4 Access control—best practices. 100

Part 3. Applications for the portal 103

Chapter 6. Integrating personal and enterprise pages. 105
6.1 Personal portal pages 105
6.1.1 Notes client 106
6.1.2 Web browser client. 112
6.2 Enterprise portal pages 122
6.2.1 Domino domain search. 122
6.2.2 Quickplace 128
6.2.3 Sametime. 128
6.2.4 News 128
6.2.5 Newsgroups 128
6.2.6 Discussion database 129
6.2.7 Library 130

Chapter 7. Adding functional and administration pages 133
7.1 Functional portal pages 133
7.1.1 Sales page 133
7.1.2 Tech support page 134
7.2 Administration pages 134
7.2.1 Portal configuration 134
7.2.2 Menu generator 138
7.3 Feedback 151
7.3.1 Designing for feedback. 152

Chapter 8. Integration using Lotus connector products 159
8.1 Overview of Domino and enterprise integration 159
8.1.1 Lotus connectors 160
8.1.2 Tools for integrating enterprise data and applications. 161
8.2 Integrating DB2 data using DECS 162
8.2.1 Activating DECS during installation 162

8.2.2 Introduction to the concepts of DECS 163
8.2.3 The sample portal customer application 165
8.3 Integrating DB2 data using LEI 172
8.3.1 The concepts of LEI 172

Chapter 9. Integration using programmatic connectors 175
9.1 Integrating DB2 data using programmatic connectors 175
9.1.1 The concepts of LCLSX 175
9.1.2 The sample portal invoice application 178
9.2 Connecting to enterprise applications with MQLSX 186
9.2.1 The MQSeries LotusScript Extension 188
9.2.2 A sample parts ordering application 190
9.3 Domino and Java 194

Chapter 10. NNTP and the Domino server 197
10.1 USENET 197
10.2 Advantages of using Domino for NNTP 199
10.3 Domino NNTP security features 200
10.4 Managing the newsfeeds 201
10.5 NNTP configuration 202
10.5.1 Setting up the server to receive newsfeeds 203
10.5.2 Setting up to serve newsgroups 208
10.5.3 Integrate newsgroups into the portal 212

Chapter 11. Collaboration 217
11.1 Sametime 217
11.1.1 Sametime overview 217
11.1.2 Installing the Sametime server on Windows 2000 218
11.1.3 Integrating Sametime center into the portal database 220
11.1.4 Using the Sametime Java applets 221
11.1.5 Integrating the Sametime chat application into the portal database 226
11.2 QuickPlace 227
11.2.1 QuickPlace overview 227
11.2.2 Installing QuickPlace on S/390 229
11.2.3 Integrating QuickPlace into the portal database 230

Appendix A. Code for the portal applications 237
A.1 Invoices application (LCLSX) 237
A.1.1 Update/insert invoice agent for Notes client access 237
A.1.2 Update/insert invoice agent for Web browser access 238
A.1.3 Delete invoice agent 240
A.2 Parts ordering application (MQLSX) 241
A.2.1 The WebMQPutOrder agent 241

A.2.2 The MQGetOrderStatus agent 244
A.3 The personal page generator 247
A.3.1 The AgentGetMailUrl agent 247
A.4 The menu generator interface. 249
A.4.1 The AddOutlineEntryWeb agent 249
A.4.2 The RemoveOutlineEntryWeb agent 250
A.4.3 The UpdateOutlineEntryWeb agent 251
A.4.4 The AddOutlineEntryNotes agent 253
A.4.5 The RemoveOutlineEntryNotes agent 255
A.4.6 The UpdateOutlineEntryNotes agent 256
A.5 The feedback application 258
A.5.1 The WebSendMail agent 258
A.5.2 The NotesSendMail agent 259
A.5.3 LiveNamesApplet Java code 259

Appendix B. Using the additional material 263
B.1 Locating the additional material on the Internet 263
B.2 Using the Web material. 263
B.2.1 System requirements for downloading the Web material 263
B.2.2 How to use the Web material 264

Appendix C. Special notices 265

Appendix D. Related publications 269
D.1 IBM Redbooks 269
D.2 IBM Redbooks collections 269
D.3 Other resources 270
D.4 Referenced Web sites 270

How to get IBM Redbooks 271
IBM Redbooks fax order form 272

Glossary 273

Index 277

IBM Redbooks review 283

Figures

1. Graphic interpretation of an enterprise information portal 3
2. Capabilities to incorporate into a basic portal design. 9
3. Our portal structure 32
4. Main portal page layout structure. 36
5. “Main Menu” outline design element 39
6. “Main Menu” form. 40
7. Enterprise portal page layout structure 41
8. Programmatic access to the Notes UI 45
9. Database settings to launch the portal. 50
10. Server setting to automatically launch the portal from a browser 52
11. Outline entry. 56
12. Defining access for the Portal group 57
13. Defining access for the Sales group 58
14. Defining access for the Tech Support group 58
15. Portal homepage for Allan Smith 59
16. Overview of the DGW localization process 62
17. DGW welcome panel 63
18. Creating the glossary for portal S/390 63
19. Choosing the template server 64
20. Creating a tagged database. 65
21. Items being tagged 66
22. Choosing a language 66
23. Selecting a language 67
24. Build language database button 67
25. Options panel. 67
26. Copy ACL from source database. 67
27. The portal homepage panel in English 68
28. Untranslated terms view 69
29. Searching for words to be translated 70
30. Marking a document translated 70
31. Marking a document approved 70
32. Glossary language view. 71
33. Re-build language database 71
34. Update language database 73
35. Outline entries for language database 73
36. Outline entry with type “action” 74
37. Hide from Web browsers 74
38. Outline entry. 75
39. Hide from Notes R4.6 75
40. Portal in French 76

© Copyright IBM Corp. 2001

41. Portal in Mexican Spanish 76
42. Notes ID password security 78
43. How basic HTTP authentication works 81
44. Establishing an HTTP session with session-based authentication 82
45. Enabling session-based authentication in the server document 84
46. Creating the Domino configuration document 85
47. Creating a Login Form Mapping in the Domino Configuration Document . 86
48. Portal login form 87
49. Web realm document 91
50. SSL handshake 93
51. Outline entries for the personal portal 106
52. Personal links page layout 108
53. Checking for subscriptions in the user settings 111
54. “GenerateWebPersonalPage” form layout 113
55. “FrameSet” layout from the personal portal page for Web environment . 114
56. Generation of personal page frameset structure at Web client 115
57. Setting up domain search on the Notes client 124
58. Activating Domino domain search 125
59. indexing time schedule for domain search indexer 125
60. Entering the server’s host name for Web client domain search. 126
61. Adding databases to the domain search 127
62. The domain search form in the enterprise portal 128
63. Configuration choice 135
64. New parameter interface 136
65. Configuration view for consults 136
66. Configuration form layout structure 137
67. ViewConfiguration view layout structure 137
68. ConfigurationPage page layout structure. 138
69. Menu Generator components 139
70. Options Menu Administration interface 140
71. FunctionalMenus form design structure. 141
72. FunctionalPage form design structure 142
73. ViewFuncPage view design structure 142
74. Functionalchoices form design structure 144
75. “ViewFuncChoose“ view design structure 144
76. Feedback panel 152
77. Adding a new image resource 153
78. Feedback outline entry. 154
79. Body text field 154
80. Hiding a field 154
81. RichTextBody field 155
82. NotesSendMail. 156
83. WebSendMail. 156

84. WebSendMail agent . . . 157
85. Domino enterprise integration solutions . . . 159
86. Selecting DECS during configuration of the Domino server . . . 163
87. How DECS works . . . 164
88. SQL to create a DB2 table . . . 165
89. Customer form . . . 166
90. Activate “Use applet in the browser” for views . . . 167
91. DECS customer connection document . . . 168
92. The DECS customer activity document . . . 169
93. Events and options in the DECS activity document . . . 170
94. DECS key initialization in the Notes database . . . 171
95. Opening a document using DECS . . . 171
96. How LEI works . . . 172
97. Relationship between different LCLSX classes with data types . . . 177
98. Searching for an invoice in the LCLSX application . . . 179
99. Updating the invoice using LCLSX technology . . . 180
100. Small frame for invoice search from Web and Notes . . . 182
101. MQLSX model . . . 187
102. MQLSX architecture placement . . . 188
103. MQLSX class model . . . 189
104. MQLSX order account management scenario . . . 191
105. Putting an order into MQSeries . . . 192
106. Viewing status change in the view after MQ get . . . 193
107. Embedding the parts ordering application in the portal . . . 194
108. Domino server NNTP process . . . 199
109. Setting up a server connection document for NNTP . . . 203
110. Setting up NNTP news feed parameters . . . 204
111. NNTP port security . . . 205
112. NNTP server preferences . . . 206
113. Automatic newsfeed downloaded from a USENET server . . . 207
114. Newsfeed types . . . 209
115. Creating a newsgroup . . . 210
116. NNTP database profile . . . 211
117. Adding newsgroup to enterprise outline menu . . . 213
118. Opening the newsgroup database from a browser . . . 214
119. Setting up an interest profile . . . 215
120. Adding the server to a Person document . . . 220
121. Adding an outline entry for the Sametime meeting Web page . . . 221
122. Entering the home URL field . . . 221
123. Creating a new Sametime application database . . . 223
124. Sametime Login form . . . 224
125. Accessing the Sametime community . . . 227
126. Opening QuickPlace . . . 231

127.Registering your QuickPlace server 232
128.QuickPlace server settings 233
129.Registering users to use QuickPlace 234
130.Specify a user directory for authentication 235
131.Creating a link to QuickPlace from the portal database 236

Preface

In today's fast-paced knowledge management environment, a great deal of importance is placed on providing professional workers with information that is tailored to their job responsibilities. This IBM Redbook gives a step-by-step description of how to build a portal application using Lotus Domino. While our sample was built in an OS/390 environment using UNIX System Services, the concepts and code samples apply equally well to any platform that supports a Domino server.

The audience for this redbook includes Domino developers, architects, and technical support engineers on all platforms, as well as consultants and marketing support professionals. A knowledge of Domino, programming, and operating systems is assumed.

The team that wrote this redbook

This redbook was produced by a team of specialists from around the world working at the International Technical Support Organization Poughkeepsie Center.

Mike Ebbers is a senior IT specialist at the International Technical Support Organization, Poughkeepsie Center. He writes extensively and teaches IBM classes worldwide on all areas of Domino on S/390. Before joining the ITSO six years ago, Mike worked for 10 years in the field in IBM/US, and for 11 years as an instructor/developer.

Armelle Creuzet is an IT Specialist in the IBM Server Group, France. She has over three years of experience in Domino for S/390 and AIX administration. She has practical experience in performing Domino for S/390 system tests, as well as several performance benchmark tests for European customer accounts. Her areas of expertise include DB2, application development in C, Java, and LotusScript programming.

Tonie Erasmus is an IT Specialist at debis IT Services Southern Africa: Technical Support. He has been in the information technology business since 1990. His areas of expertise include application development, Internet development, systems programming and consulting on IBM S/390. He specializes in S/390 UNIX System Services and e-business integration with S/390 enterprise data and transaction systems.

Rudolf Fust is an IT Specialist in IBM Global Services, Germany. He studied Computer Sciences at the University of Hamburg. He has over five years of

© Copyright IBM Corp. 2001

experience in Domino application development and administration. He is a Certified Lotus Professional at the Principal level in Domino R4 and R5. He gained practical experience in several large application development projects at customer accounts. His areas of expertise include DB2, DECS, LEI, and Lotus connectors.

Ernesto Ramos is an IT Consultant for Ddemesis in Mexico. He studied Applied Mathematics and Computation at the National Autonomous University of Mexico (UNAM). He has over five years of experience in Domino application development and project management, and 12 years of experience developing in diverse programming languages such as C, Visual Basic, Delphi, Cobol, and Pascal. He is a Certified Lotus Professional in Domino R5, and a Certified Lotus Professional at the Principal level in Domino R4. He has practical experience from several large application development projects. His areas of expertise include Add-Formula, LotusScript, JavaScript, Java, and HTML.

Erna Schmidt is a Senior Domino Specialist working for T-Systems in Cape Town, South Africa. She has over six years of experience in Lotus Notes and Domino, including development and administration. Her areas of expertise include infrastructure design and implementation, installation, upgrades, support, and integration with Exchange and other third party products. She has 11 years of IT of experience, including mainframe programming, Roscoe administration, Sybase and SQL Windows, and project management.

Thanks to the following people for their contributions to this project:

Richard Conway
Robert Haimowitz
Michael B. Schwartz
International Technical Support Organization, Poughkeepsie Center

Phil Kalinowsky
IBM Poughkeepsie

A special thanks toes to the authors of the redpaper *Building a Portal with Lotus Domino R5,* REDP0019. It applies to all platforms and is available from the IBM Web site `http://www.ibm.com/redbooks`. The authors are:

David Morrison, ITSO, Cambridge Center
Martin Buckley, Lotus Corporation, Australia
Steve Cappo, Lotus Development, Cambridge

Comments welcome

Your comments are important to us!

We want our redbooks to be as helpful as possible. Please send us your comments about this or other redbooks in one of the following ways:

- Fax the evaluation form found in “IBM Redbooks review” on page 283 to the fax number shown on the form.
- Use the online evaluation form found at **ibm.com**/redbooks
- Send your comments in an Internet note to redbook@us.ibm.com

Part 1. Introduction to S/390 and portal concepts

© Copyright IBM Corp. 2001

Chapter 1. Portal concepts

Portals are applications that enable companies to unlock internally and externally stored information, and provide users a single gateway to personalized information needed to make informed business decisions. They are an amalgamation of software applications that consolidate, manage, analyze and distribute information across and outside of an enterprise (including Business Intelligence, Content Management, Data Warehouse and Data Management applications.).

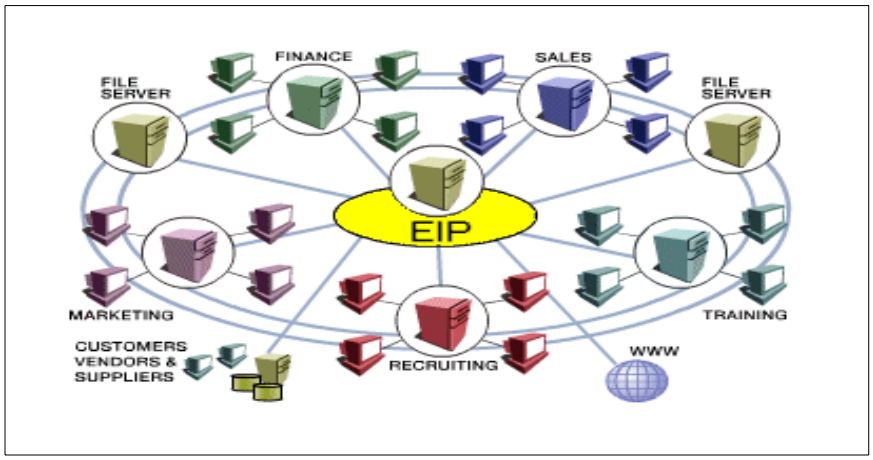


Figure 1. Graphic interpretation of an enterprise information portal

Much has been spoken and written about the concept of a portal. Today it is considered one of the most important applications a company can have.

But why? What is a portal and why do we need one? A portal is a tool for managing a company's intellectual assets, information content, and data. It takes this conglomeration of information and organizes it, categorizes it, personalizes it, and presents it to each user at the right time and in the right context.

For example, if you want information on one of your clients, you should be able to see all the relevant data in one location. You should not have to search through multiple systems such as an accounting package or a contact manager, or sift through paper documents and your file server just to find what you need. It should be organized and presented to you when it is most appropriate.

© Copyright IBM Corp. 2001

A portal is also an opportunity to streamline access to information. As you can imagine, organizing information, streamlining processes, reducing printed output, and saving time all have a significant impact on the effectiveness of an organization, and can provide a significant cost reduction. Companies typically go to great lengths to manage the information stored within their databases (structured data from accounting systems, customer lists, and so forth), but they tend to ignore the data that exists in non-structured formats (such as documents).

1.1 Different kinds of portals

It has been said that defining a portal is a political activity. That is, different vendors and users could suggest definitions that suit their own purposes and analytical processes. Let us look at a few types, for discussion purposes.

1.1.1 Horizontal versus vertical portals

First we contrast horizontal and vertical portals.

1.1.1.1 Horizontal portals

Most people are familiar with general-purpose portals, which are classified as horizontal portals or *megaportals*. Ever since1996, these portals have been used to catalog available content from the Internet, acting as hubs from which users could locate and link content. The business model for this type of portal consisted of selling advertising banner space and successfully directing Web surfers to their desired destinations to ensure repeat business.

Portals now offer a broad array of online resources and services. Although no single model defines a portal, all portals offer at least four core features: Web searching, news, reference tools, and some communication capabilities (such as e-mail and chat).

1.1.1.2 Vertical portals

Vertical portals are the fastest growing type of Internet portal.

Vertical Internet portals differ from horizontal portals in the range of users that they target. Horizontal Internet portals try to service the entire Internet community, while vertical portals target a niche audience—usually a particular industry. Vertical portals provide content relevant to their industry, with links to industry, supplier and even competitor sites.

A vertical portal has community and collaboration capabilities that are aimed at its industry or niche, and e-commerce services for products and services relevant to its industry or niche. Vertical portals also try to leverage portal

concepts such as branding and associated technologies, but aim to do so in a more focused way (for example by concentrating on a geography, a specific industry, or a special interest).

Vertical portals are gaining popularity, particularly in the business-to-business, business-to-consumer, and business-to-employee space. We will use the business-to-employee concept to define the corporate portal.

1.1.2 Corporate portal

Corporate portals must connect the users not only with everything they need, but with everyone they need, as well as provide all the tools they need to work together. This means that groupware, e-mail, workflow, and desktop applications, even critical business applications, must all be accessible through the portal.

One can distinguish at least four sub-types of corporate portals:

- *Information* portals connect people with information by organizing large collections of content on the basis of subjects or themes.
- *Collaborative* portals enable teams of users to establish virtual project areas or communities along with the tools for collaboration, and to work cooperatively within these communities.
- *Expertise* portals link people together based on their skills and expertise, as well as their information needs.
- *Knowledge* portals combine all of the above to deliver personalized content based on what each user is actually doing.

A corporate portal could differ from an Internet portal in that it can:

- Integrate access in a wider variety of data formats than a web portal (comprehensive).
- Organize access to information for users to browse (organized).
- Assemble personalized views of key information and notifying users of the availability of new material via electronic mail and other media (personalized).
- Organize access to data, but not necessarily store the data itself (location-transparent).
- Support extensions for cataloging new types of information (extensible).
- Automatically identify and organize access to new content (automated).
- Selectively broker access to internal corporate information (secure).

This definition is clearly oriented toward distinguishing corporate portals from public portals on the basis of the kind of access available in corporate portals. It is not focused on the types of applications supported by such access, however, and is consistent with decision processing portals with or without collaboration, collaborative/work flow processing portals, expertise processing portals, and knowledge portals.

Whatever the political issues involved in defining a portal, the reality should be that a successful portal must offer strategic advantage to the community it seeks to target.

1.2 Key features of a portal

Categorized below are many of the features one typically finds in a portal:

Personalization

- Preferences
- Push-pull based information (subscription)
- Scheduling events
- Setup of filters
- Personal profiles
- Customizing
- Proactive help
- Suggestion learning

Categorization

- Organizing taxonomy
- Index-crawl
- Auto-classify summarize
- Clustering
- Extraction
- Feeds
- Metadata manager
- Content mining
- Full text search
- Connection to structured versus unstructured data

- Business intelligence
- Connection to external content

Applications

- Application components
- Line of Business applications
- Horizontal applications
- Custom applications
- Collaboration
- Community knowledge exploitation
- Asynchronous personal information manager
- Synchronous personal information manager
- Centralized repositories
- Distance learning
- Transaction
- Transaction management
- Workflow

Common core portal services

- Single sign-on
- Directory integration
- Security
- Access control rights
- Secured delivery
- Authentication
- Permissions
- Administration

This list can never be completed, since the portal space is evolving rapidly. For example, the integration of pervasive devices, such as the cell phone and PDAs using WAP or Bluetooth, is already becoming a necessary function of a portal.

1.3 So what makes a portal a portal

In this redbook, we follow the conceptual model used in the redpaper *Building a Portal with Lotus Domino R5* by David Morrison, Martin Buckley and Steve Cappo. Their description of a portal incorporates most of the features described in the previous section, and provides a clear model to the layman on how to start building a portal.

According to this paper, “a portal is an application or device that provides a personalized and adaptive interface for people to discover, track, and interact with other relevant people, applications, and content.”

Incorporating most of the features listed in 1.2, “Key features of a portal” on page 6, they summarize them as follows:

1. Personalization for end users is the most critical feature. A portal must deliver a personal or community desktop to users by establishing unique looks, content, and application interfaces, and proactively rendering them based on the user’s role in their community or by actively tracking the user’s individual usage, interests, and behaviors.
2. Organization of the user’s desktop to eliminate the information glut. Users want consolidated access to their important contacts, applications and content. The concept of stovepipe applications are a thing of the past. Organizations want easier control to design their desktop in a layout that suits them.
3. Resource division determines who sees what. Portals must have a membership services layer for user authentication, single-logon, and credential mapping. Users demand the highest level of security, but the least amount of annoyance.
4. Tracking of activity provides users with a payback for using the portal. The more users use the portal, the more it becomes tailored to specific interests and affinities the user may develop. While this may sound threatening at first, users will have the ultimate control over what gets tracked.
5. Access and display of aggregated multiple heterogeneous data stores, including relational databases, multidimensional databases, document management systems, e-mail systems, Web servers, news feeds, and various file systems/servers (audio, video, image, and so on). It is extremely helpful for users to see their e-mail next to news feeds and a list of online users who can help understand information while maintaining a single context.

6. Location of important people and things. A portal is based on the basic desire of users to easily find information and people by searching or navigation. There must be a means of passively (or actively) discovering the experts, communities and content in a relevant context.

Figure 2 shows a starting model for building a basic portal.

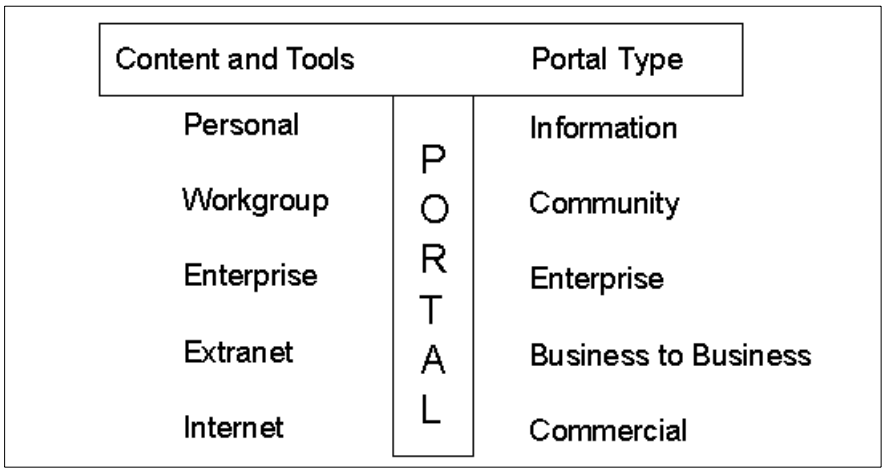


Figure 2. Capabilities to incorporate into a basic portal design

1.4 Benefits, elements, and first steps

By this time, the benefits of a portal are probably obvious to you. However, we include a list for the sake of completeness.

1.4.1 Benefits

- Users receive information through a standardized Web-based interface.
- Users can interact, question and share information from their desktops.
- Portals integrate disparate applications and other data external to these applications into a single system that can share, manage, and maintain information from one central user interface.
- A portal is able to access both external and internal sources of data and information.

1.4.2 Elements of knowledge management

Portals deliver on the promise of knowledge management by providing end-users with a single point of access to the *people, places, and things* that

exist within and across their company boundaries. Through the product's place-based design, end-users not only access the presentations, documents, discussions, task lists, and countless other forms of business material that they need to complete their tasks, but also have a forum for acting on that material and engaging others in their efforts.

People: People are among the most valuable assets a company can possess. People retain critical business knowledge—including knowledge of processes, contacts, and resources—that can often make the difference between success and failure on any given business project. Online “people awareness” and real-time communication tools, such as the ones provided in portals, offer a means for the end-user to view the status and online location of other users. Moreover, these tools provide a means to engage these users in instant online communications through an easy-to-learn and easy-to-use interface.

Places: Places provide a context for work activities. Just like a physical office space, an online workspace needs to be well equipped and well organized so that people can focus on their task and not the tools needed to achieve it. Portals provision their online workspaces with out-of-the-box templates that can be used to immediately design and implement online workplaces. Additionally, application developers can create new templates to meet virtually any business need and can customize details such as their look and access status. When instituted, places make it easy for non-technical end users to complete their business tasks and interact with others. The result of their efforts—perhaps a business plan or an RFP—then becomes a *reusable knowledge object* that can be accessed by others who may want to undertake similar work. Therefore, places not only provide the context for the development of knowledge, but also a means for its preservation and ultimate reuse. This is a powerful capability in fast-changing and highly competitive markets.

Things: Things are the essential resources or content that people access and utilize to perform business tasks. The spreadsheet, the document, the Web site, are all things that are used in daily business activities. With portals, these various types of content can be displayed and arranged in a meaningful context. In particular, users can arrange things in their Personal and Community Places so that they can improve the effectiveness of their personal- and group-based efforts. In both cases, content-bearing portlets can be built from templates or custom designs and can be easily re-sized and moved through drag-and-drop maneuvers.

People, places and things should be the fundamental components of a portal solution. By uniting people through real-time technologies, portals should

improve the speed and accuracy of business decisions. Moreover, through collaboration features, communities can assemble, collaborate, and ultimately innovate across geographical boundaries—all from a single point of access.

1.4.3 First steps

The first step in building a portal is planning. A portal should be structured in such a way that it allow users to:

- Discover information quickly instead of searching for it
- Have consolidated access to applications
- Be part of a community
- Have access to critical resources and information relevant to their job
- Have a uniquely personalized interface

If you map these five functions across the following content areas, you have completed the first step in planning and building a portal:

- Personal content
- Workgroup content
- Enterprise content
- Extranet content
- Internet content and tools

The information in each of these content areas can be personalized for each user.

Chapter 2. Why isn't your Domino server on a S/390?

Companies around the world are calling on the powerful functionality of Lotus Domino technology to handle many of their critical system needs. It is very common to see Domino applications for enterprise messaging, groupware, intranet/extranet support, and e-business (Web) processes.

As these applications become critical components for a company's infrastructure, we must recommend our best combination of hardware and software to support them. This must cover the requirements of reliability, availability, scalability, performance, security, serviceability, and manageability. The solution must accommodate the "Grow-As-You-Go" capabilities provided by Lotus Domino, allowing the company to invest in the future of its business.

For most of the hardware and software combinations that support Domino applications, it is very difficult to cover these requirements. In fact, some of them don't meet all the needs with the efficiency and quality required for these strategic applications.

We recommend the implementation of Lotus Domino on S/390 systems as the best combination to meet these requirements. In addition to having the capabilities of all other platforms, S/390 helps cover many special needs, such as supporting the deployment increases and process simplifications that occur as a result of the natural growth and changes in the business processes.

Lotus Domino on a S/390 helps to:

- Reduce total cost of ownership
- Reduce personnel cost
- Increase user satisfaction and productivity through increased reliability, availability, and overall service performance (RAS)
- Grow non-disruptively (scaleability)
- Reduce complexity through consolidation
- Provide central control of a distributed infrastructure
- Provide dependable back-up and recovery
- Provide bullet-proof security
- Facilitate enterprise integration with back-end data
- Add the potential of consolidation on the order of 40 servers to 1

© Copyright IBM Corp. 2001

The S/390 is a very competitive option for Domino deployment for a wide range of organization sizes, industries, and application requirements. In large companies, moving to S/390 has tended to coincide with IT reorganizations that have created centralized management infrastructures for key services. Companies wanted to achieve enterprise-wide integration of key systems, and to realize economies of scale.

The excellent cost/benefit ratios, along with the high levels of scalability and I/O performance of the zSeries architecture, make it a very attractive option for hosting large-scale, high-volume Lotus Domino solutions extending beyond the enterprise. To get a detailed analysis of cost/benefits, see the article “Enterprise solutions for Lotus Domino: Cost/Benefit Analysis for Large-Scale IT Deployment” at this Web site:

`http://www.s390.ibm.com/ftp/os390/domino/itg_paper.pdf`

The following sections provide an overview of the benefits we can obtain using the powerful combination of Lotus Domino and S/390 systems.

2.1 Domino in a S/390 environment

The S/390 environment is an ideal complement to Lotus Domino applications. With the combination of these technologies, you are able to get powerful applications running on a robust and mature platform, efficiently meeting our most critical requirements of reliability, availability, scalability, performance, security, serviceability, and manageability. All this, and still we can achieve the best cost/benefit ratios currently in the marketplace. These characteristics make Domino for S/390 a highly valuable solution.

The Domino server on S/390 uses OS/390 UNIX System Services, so you access the S/390 Domino server using a UNIX 95-compliant interface. Users access a Domino server on S/390 in the same way as on any other platform, using a Notes client or a Web browser. They are able to enjoy the same functionality and “look and feel.” The users cannot tell which platform their Domino server is on.

Also, you can manage the administration of the Domino server using a Notes administration client from any Notes workstation in the same way you do it in any other platform. One difference is that Domino administrators are not able to start the S/390 Domino server from the Notes administration client; this is the responsibility of the S/390 console operators.

The S/390 is a mature and powerful server platform that provides a lot of functionality to support Domino applications. One of the most valuable

advantages offered by S/390 is the ability to use a large Domino server to support many Notes applications without needing to implement each Notes application on its own dedicated server.

Using a single server helps to reduce the administration effort required to manage a Notes environment. This is directly reflected in a substantial reduction in the cost incurred from this essential activity. You also have more efficient access to current data in a particular application, eliminating the delay time required to propagate the updates from a database (using the replication function) to the multiple copies distributed along a network engine integrated by multiple small servers. You can achieve faster mail routing by eliminating the time required to route mail across a network of small servers. With the reduction of replicas of a particular database, you also reduce the use of disk space dedicated to them, and with this, you also reduce the network traffic.

The following section presents a more detailed explanation of the advantages of using the Domino server on the S/390 server platform. We include an excerpt from chapter 1 of *Lotus Domino for S/390 Release 5: Installation, Customization and Administration,* SG24-2083.

2.2 The value of a large Domino server

So far, many Notes rollouts have used a large number of relatively small servers, because this was the only option available. Organizations that have done this have found that there are a number of disadvantages in this approach.

S/390 has the capability to support a large Domino workload. In July 1999 we announced Notes bench results that certified support of 32,000 connected mail users, making it the largest Domino server available. See `http://www.notesbench.org`. In addition to these test results, consider S/390's track record:

- IBM has been running a production Domino server on S/390 since November 1996. This supports the largest number of users on any Domino server in production use within IBM, and the largest number of active Notes client users in the world.
- The S/390 platform is easily capable of supporting many thousands of users. Many S/390 customers do that today with other applications. We know that the S/390 hardware and software can run production systems with thousands of users because we have been doing it for many years.

A S/390 server is able to support large workloads because IBM has invested in many facilities in the processor and software to provide scalability, including:

- Very efficient use of symmetric multiprocessors (SMPs), even up to 12-engine machines
- Addressing of 4 terabytes of expanded processor storage along with the 31-bit (2 GB) storage addressing that it has had for many years
- Efficient techniques for keeping key data in processor memory where it can be accessed quickly
- A sophisticated job scheduler
- The removal of bottlenecks
- Shared access to all disks by all processors
- Separate, dedicated I/O processors

So what are the benefits of a large server for Domino?

2.2.1 Less administration effort

The larger the server, the fewer the number of servers you need to run. Fewer servers mean less administration effort to manage your Notes environment. This is a major issue, since surveys show that more than 80 percent of the cost of a client/server solution is in administering a system. It is also an area of cost that is easily forgotten when you start the project, but which becomes a major focus once you roll out many servers and need to provide good service.

Fewer servers mean less work in the following areas:

- Installing, monitoring, and repairing hardware
- Installing and upgrading operating systems and Domino software
- Installing and managing Domino servers and the connections between them
- Monitoring and managing inter-server tasks, such as mail routing and replication

If you can host an application, or even a whole domain, on a single server, many design tasks become easier. You do not need database replication to other servers. You do not need mail routing, Public Address Book distribution, or network connections. The best way to reduce work is to eliminate it completely.

2.2.2 Access to current data

When an application is distributed across multiple servers, the various copies of the databases are kept in sync using replication. However, replication is an asynchronous process, which means that, at any point in time, the most current data will not have been propagated to all of the replicas.

In the ideal case of a single server for an application, there is no replication and everyone accesses the single up-to-date copy of the data. In an environment with multiple servers, the fewer the servers the more current the data.

2.2.3 Faster mail routing

When mail is sent from a client, it is stored in the server's mailbox and then forwarded to the target server by the router task. It takes time for the mail to be routed across the network. While a server can route mail to several servers at the same time, it only opens one connection to each server. Therefore, if a large mail file is being transferred to any server, all other mail destined for the same server must wait until the first transmission finishes.

If the mail files for many users are on a single server, mail between them does not need to be routed across the network. Mail is placed directly into the recipients' mail file, which is much faster.

2.2.4 Less disk space

When an application is spread across multiple servers, the design usually involves multiple replicas of each database. These all use disk space. By reducing the number of servers, you reduce the number of replicas. This can save huge amounts of disk space.

2.2.5 Less network traffic

Regardless of how many servers you have, the amount of client-to-server network traffic will be the same (although with centralized servers that traffic may need to go over a wide area network, while it could go over a LAN if the server is local). If you have multiple servers, you also have inter-server traffic for database replications, mail routing and access to other applications. The more servers you have, the more inter-server traffic you will have. Therefore, by reducing the number of servers you can reduce the total amount of network traffic.

2.2.6 Less processor capacity

If you run multiple servers, you must install sufficient processor capacity on each server to handle its peak usage. It is very likely that the peak usage of each server will be at a different time and, therefore, at any one time you will have a lot of unused capacity in the overall system.

As an extreme example, if you were to place servers around the world to support users in many countries, then at any one time some servers would be busy and others would be idle because it is the middle of the night in that time zone. By consolidating all of the users onto one server, you do not need the total capacity that your smaller servers had, so your server capacity could be much smaller.

In practice, you can reduce the total server capacity you need even by consolidating servers in a single location. One S/390 user found that total capacity could be reduced by 22 percent just by consolidating two servers in the same location onto one.

2.3 Other unique S/390 benefits

We saw previously that a large server has many benefits for Domino, and S/390 is a large server. However, S/390 has other unique benefits, which we discuss in the following sections. There are good reasons why many of the world's largest banks, insurance companies, government agencies, and other large enterprises use S/390 to deliver applications.

A more detailed discussion of this subject is available in *Selecting a Server—The Value of S/390*, SG24-4812.

2.3.1 Multiple applications on one server

The S/390 server runs many applications at the same time. Since today's view of a server is a box that runs one application, this is one area where S/390 may be *contrary* to your expectations of a server platform.

The philosophy in the UNIX and PC server marketplace is to run each application on its own server. In a Notes deployment, the recommendation is to split up the functions so that each function runs on a dedicated server. The reason for this is to provide workload isolation, security, availability and performance. The underlying reason is that the operating systems and hardware on these platforms are not able to isolate the effects of multiple workloads on the same server; that is, the failure of one application or function could cause a failure of the whole system, or one application could

take more than its fair share of system resources, causing poor performance of other applications.

This is *contrary* to the S/390 philosophy. Since the early 1970s, our customers have told us that they want to run multiple applications on a single server. We therefore built into the hardware and software the functions that you need to do that and still provide excellent workload isolation, security, availability and performance. For example:

- OS/390 virtually never fails. More than 50 percent of the OS/390 kernel code, and much of the S/390 hardware, is concerned with detecting errors and resolving them while keeping the system running. One customer ran for more than 2,000 days without “rebooting” (re-IPLing).
- If one application fails, the operating system isolates it to the smallest possible part, cleans it up and allows it to be restarted. Other applications are not affected.
- OS/390 has a very sophisticated workload manager that ensures that each application gets only its fair share of the resources, according to its needs. These needs may change at different times of the day, and OS/390 will continuously rebalance priorities.
- OS/390 isolates users and applications from each other.Most users of S/390 servers run many applications on a single server. You should consider this normal for a S/390 server.

2.3.2 Running Domino alongside enterprise applications

Running Domino alongside other enterprise applications on a S/390 server has many benefits:

- It further minimizes your administration effort because you have fewer platform types to support.
- It further reduces the total processor capacity that you need, especially because different applications are likely to have their peak loads at different times.
- It allows applications to communicate at processor speed, without going across a network connection.
- A lot of enterprise data is stored on S/390 servers. Many Notes users wish to access that data and the applications that process it. By putting the Notes application on the same physical server, you accomplish the following:
  - Improved performance
  - Improved availability

- Simplified system management
- Minimized server and network costs
- Eliminate the need to take copies of the data onto other servers, which takes time and resources and means that you may be working with back-level data

Even if you wish to support only a small number of Notes users, there are advantages to running the Domino server on a S/390 alongside other applications rather than investing in a separate server with its own support needs.

2.3.3 Multiple operating systems on one server

S/390 goes a step further and allows you to run up to 15 different operating systems on a single S/390 server. This function is called Processor Resource/Systems Manager (PR/SM), also commonly referred to as *logical partitioning*. PR/SM provides an additional alternative to Domino partitioning. PR/SM provides a higher degree of isolation by running separate operating systems, although at a slightly higher cost in terms of system resources used. PR/SM is part of the strategy to minimize the number of servers, and it is used for the following:

Hardware consolidation

You can run different operating systems or operating environments on a single server. This could be different operating systems, such as VM, OS/390 and TPF, or different versions of OS/390, for example. You can use this to consolidate multiple servers, including Domino servers, onto one physical server without merging the environments. This is a quick way to reduce the number of servers you run.

Development and test

You can provide a separate development and test environment (and Notes domain) on the same physical server for the upgrading of operating system software, Domino server code, or testing new functions and applications.

Workload isolation

You may need to keep some workloads separate; for example, if you provide outsourcing services to clients, to provide security between workloads (for example, separating internal and external communications), or if some workloads require extremely high availability.

2.3.4 Low costs

Many surveys by independent consultants have shown that S/390 is a less costly solution than running many small servers. See, for example:

- *Cost of Messaging: Comparative Study of the S/390 - Lotus Domino Solution*, by the International Technology Group
- *Domino for S/390 - The Business Case*, by the International Technology Group
- *Selecting a Server - The Value of S/390*, SG24-4812.

S/390 allows you to run a small number of large servers and provides you with advanced automation tools. These dramatically reduce the costs you will incur in supporting your environment.

Be careful when you estimate the costs for other server solutions. If you do not have experience in running these types of systems, it is very easy to underestimate the costs of hardware, software, and support staff to get good availability and performance in those environments.

2.3.5 Security and integrity

A major concern of commercial and government enterprises is the security and integrity of their data. This concern is magnified when connections to the Internet, or even extended access through an intranet, are considered.

OS/390 provides a strong framework for control of data access. Security is controlled by the System Authorization Facility (SAF), which checks all access requests to system resources using an external security manager such as the OS/390 Security Server (formerly known as RACF). OS/390 is committed to maintaining the integrity of your data, whatever happens. IBM announced an integrity guarantee for MVS in 1972, and S/390 is designed to ensure that data integrity is carefully maintained.

It is also much easier to provide physical security for a centralized S/390 server than for many distributed servers.

2.3.6 Reliability and availability

S/390 systems are used in environments where high availability is crucial. S/390 hardware and software have many availability features built in. These include the avoidance of, or automatic recovery from failures; and the ability to apply system maintenance with minimal disruption to system operation. As a result, many S/390 users run a single S/390 server and get excellent system availability. In contrast, many UNIX and PC servers need clustered

systems, at additional hardware and software cost, to achieve similar levels of availability.

Keep in mind that availability is a key requirement for any application that uses the Web serving function of Domino. The Web culture has caused us to become very impatient consumers of information. If a particular Web site is not available or is responding slowly, the client will shift his/her attention to another site, perhaps that of a competitor. The Web also extends the hours of opening for your enterprise to 24 hours a day since clients can access at any time of the day or night, or may even be located across the world.

2.3.7 Growth and flexibility

Suppose that you wish to grow your Notes environment by adding new users and applications. We have already seen that growth can mean adding many more servers. It could also mean upgrading servers or splitting users across more servers. Do not underestimate the amount of effort in doing this. It can be a major logistical exercise, especially if the servers are distributed geographically. It will take time and effort to do it, and the effort to manage your new Notes environment will increase significantly.

In contrast, upgrading a S/390 server is easy. It can often be done in a few hours, usually with no associated software changes, and the upgrade does not significantly increase the amount of system administration required. The more you grow, the more efficient S/390 becomes.

There are many different sizes of S/390 servers, from a server based on a PC (P/390) to a cluster of very large processors in a Parallel Sysplex. When choosing your initial server, keep in mind the size of your eventual implementation and select a server that can be upgraded to meet your future requirements.

Consider also a requirement to upgrade the Domino server code, which may require an upgrade to the operating system software. With a single S/390 server you only need to do that once, and as a S/390 user you will have experience with doing operating system upgrades. Contrast that with needing to upgrade the software and Domino code on many servers that may be geographically remote.

Your server choice can have a major impact on your future business flexibility. Suppose you connect a Notes application to the Web and the access rate goes up by a factor of 10 or 100. How will you handle that kind of growth? If you cannot handle it quickly, you will significantly impact your customer service.

2.3.8 Powerful I/O capability and backup speed

One of the factors that is often forgotten until it causes problems is the speed at which a server can process data. This may not be a problem during the early stages of a project, but in a production situation it becomes critical. Certain tasks on the system, including replication and taking data backups, require the processing of large amounts of data. If this cannot be done in the available time frame, it can have a major impact on your applications.

The S/390 server has a very powerful I/O subsystem. The I/O processing is done by a separate set of dedicated processors within the S/390 server. Therefore, S/390 servers can do I/O intensive tasks such as backup and recovery of data much faster than many other server platforms.

2.3.9 Data management

S/390 servers have the capability to support large amounts of data. There are S/390 servers in production today that access more than one terabyte of data. S/390 has the speed to access that data, and the tools to manage it.

2.3.10 Low risk

In any project there is a risk of failure. We have seen many client/server projects fail over the past five years, in general because the risks of the project were not accurately assessed and managed. You need to ask what could go wrong, account for it in your initial choices, and manage the risks throughout the project.

In many cases, users are deploying large systems that they have no previous experience with. As they go through the project and better understand the capabilities of the various components, they often realize that they need to buy a lot more hardware and software, put in many more support staff, or that the technology is incapable of delivering what they expected. These items should have been identified at the start of the project, risk factors built into the costs for items not clearly understood, and key areas investigated at an early stage.

In today's fast-moving marketplace no server solution is going to be risk-free. Different solutions will have different risks associated with them. However, S/390 does have one major advantage: many users have a lot of experience in running large commercial systems on S/390 servers. We know what you need to run a large production system using S/390 servers, in terms of hardware, software and support staff. That has sometimes been to S/390's disadvantage since the costs of a S/390 server are visible and known. However, it is a major advantage when you look at risks. S/390 is a lower risk

solution. You are not likely to get many surprises with a S/390 implementation.

2.4 Developing Notes applications on a S/390 platform

Domino can be used to develop and deliver a wide range of applications. For example, if you want to integrate new Domino applications with existing applications and data that reside on S/390, you can do this using enterprise integration connector products. Running Domino applications on the same server as the existing applications and data improves the performance, accessibility, and manageability of the total application system, and you can put a new front end on existing applications. For instance, you could Web-enable existing systems using Domino as the development and delivery mechanism for the new user interface.

Let us turn to some of the considerations when we install a Domino application on a S/390 server.

2.4.1 Gathering the prerequisites

Here are the main items that you need to run Domino on S/390. For a complete list, see the Domino for S390 installation guide at:

`http:www.notes.net/doc.`

Lotus Domino for S/390 Release 5.0.5 requires the following software and hardware:

- Minimum requirement is OS/390 Version 2 Release 6 or higher and the required PTFs listed. OS/390 Version 2 Release 7 or higher and the required PTFs listed is preferred due to the significant performance improvements in DFSMS.
- Processors supported: Any processor that supports your release level of OS/390. See the note later in this section.
- Java Development Kit (JDK) 1.1.8 at the July 21, 2000 service level. Please ensure you have applied all of the JDK prerequisite PTFs. Do not use JDK 1.2 (Java2) with any S/390 5.0.x release of Domino.
- The OS/390 C/C++ IBM Open Class Library installed.
- No license for the C/C++ feature of OS/390 is required.
- SCLBDLL must be in the program search order (for example, in the SYS1.PARMLIB member, LNKLSTxx or PROGxx).

- A workstation with a CD-ROM and a connection to the S/390 where Domino will be installed.
- TCP/IP networking support.
- DASD volumes for the HFS datasets where Domino data will reside.

Note

For Domino applications that have frequent floating point operations, an S/390 processor with IEEE floating point hardware (G5 or G6 processor) is recommended.

The following Web site contains the list of OS/390 services required to run Domino for S/390:

`http://www.s390.ibm.com/products/domino/servchoice.html`

Additionally, a tool (the PTF checker) is provided that uses SMP/E to verify that this service has been applied on an OS/390 system. The PTF checker is also provided at the Web site listed above. To use the PTF checker, follow the directions in the comments section of the Domino for S/390 installation manual.

The Web site should be reviewed frequently and service should be applied and verified whenever there are changes to the list.

2.4.2 Creating a support team

Because S/390 servers run multiple applications, management of the server is done by a central group of operators and system programmers. Administration of the S/390 server and operating system is done only by the S/390 support staff, to protect the overall system. Notes administrators do not need to manage the server hardware and software, but they have to work with the S/390 support staff to ensure that they get the environment that Domino needs.

it is important that S/390 operators understand the importance of the Domino server to the organization, and that they have some understanding of Notes so that they can provide better support to the Notes administrators. Therefore, consider including key representatives from the S/390 operations and support areas in the initial rollout. They will better understand the impact of some of the things that they control, such as performance and availability, on the Domino environment.

Administration of the Notes environment is done by Notes administrators, as for other Domino servers. S/390 support staff can think of Notes as a major subsystem on OS/390, similar to CICS, IMS or DB2, with its own specialist support staff.

It is critical to the success of the Notes deployment on S/390 that the two groups of people work together as a team, that they build an overall support plan for Domino on S/390, and that everyone understands the specific roles of each group and how they interface with each other.

2.4.3 Service strategies

There is one service strategy that is consistently effective: determine the OS/390 fixes for your releases of Domino, then install all program temporary fixes (PTFs) on this list (whether they seem to apply or not). Domino uses a wide variety of OS/390 system services, so if it is on the PTF list, it needs to be applied.

Lotus regularly provides updates to Domino that contain new function as well as maintenance. Unlike the IBM maintenance procedure, which updates just small portions of product code, these updates are a complete replacement of the Domino product code. A quarterly maintenance release (QMR) is issued about every three months (for example, Domino 5.05). If there are “hot” fixes that need to be issued sooner, they come in a quarterly maintenance update (QMU, for example Domino R4.04a). There are also incremental updates that allow upgrades from one QMR to another, without the need to install the entire code stream.

Visit the home page of Domino for S/390 at URL:

`http://www.s390.ibm.com/domino`

For a complete guide to the Domino server on a S/390 platform, see *Lotus Domino for S/390 Release 5: Installation, Customization and Administration*, SG24-2083.

2.4.4 Writing the portal

With all of the above information, using Domino for S390 to host a portal (for internal or external users) is an obvious choice.

The rest of this redbook explains how to design and build a portal and how to include applications in your portal architecture. We also describe how to connect to the enterprise data sitting on the S/390 server from your Domino server.

Part 2. Creating the portal architecture

© Copyright IBM Corp. 2001

Chapter 3. Kernel portal architecture

This chapter explains how to design and create the kernel part of the portal. We discuss the portal structure and layout and then show how to build the kernel.

3.1 Individualized levels of content

A portal application organizes, structures, filters, and captures information that would otherwise be less valuable to a specific user. Therefore, a portal should be structured in such a way that it targets specific information and applications. The goal is for users to spend less time wading though irrelevant information and using different interfaces to complete their tasks.

3.1.1 Levels of content

Here are the levels of content that an enterprise portal can provide:

- Personal content
- Workgroup content
- Enterprise content
- Extranet content
- Internet content

3.1.1.1 Personal content

Personal content is related to information and applications for an individual user. This could include:

- E-mail
- Calendaring and scheduling of both people and resources
- Workflow processes for the specific user
- Personal addresses

3.1.1.2 Workgroup content

This is related to information and applications oriented to a user's specific workgroup or job function. This content is usually organized by department or job, such as sales, human resources, finance, marketing, or information systems. Examples of workgroup content are:

- Project team documents, task and team spaces
- Sales tracking or relationship management applications

© Copyright IBM Corp. 2001

- Workgroup-specific document libraries or discussions
- Product information
- Workgroup- or function-related applications

As users generally spend most of their time in their workgroup, this content area requires the most thought and planning. Questions to consider are:

- How is your organization split up?
- How can you separate information, resources, and applications by workgroup?
- Is your organization structured horizontally (sales, marketing, human resources) or vertically (manufacturing, telecommunication, banking and finance)?

Content of a workgroup is generally not made available to users outside the workgroup because the content is only relevant to members.

3.1.1.3 Enterprise content

This is any information or application that spans the whole organization and is used by most employees. Examples of enterprise content are:

- Human resource information (polices and procedures)
- Vacation processing
- Travel requests
- Help desks
- Enterprise news

This information usually consists of content or applications that already exist in your organization. We can utilize this information by linking it into an area of the portal where everyone has quick access.

3.1.1.4 Extranet content

Extranet content is interactive communication between a company and selected external business outside of an organization over a secure network. This could be:

- Supplier order information and applications
- Shared projects with team members outside your company
- Shared business partner applications

This content is not always used by everyone. Usually only one or two workgroups will use an extranet application.

3.1.1.5 Internet content

This content generally consists of information or tools available on the Internet, such as:

- Customer Web sites
- Stock and weather information
- Subscription news services

This content may be made available to certain workgroups or to everyone.

3.2 Designing the kernel portal application

We used the versatility offered by Lotus Domino to create applications that work in both Notes clients and Web browser clients. This advantage helped us to create a powerful and flexible portal enterprise application that is able to cover the requirements the users have for both environments.

When we designed our portal, we took special care to preserve the same look and functionality for both environments. Thus the users do not notice the difference when they are working in either the Notes client or Web browser environment. With this characteristic, we eliminated the necessity for users to learn two different ways to complete the same kind of activities.

3.2.1 Portal structure

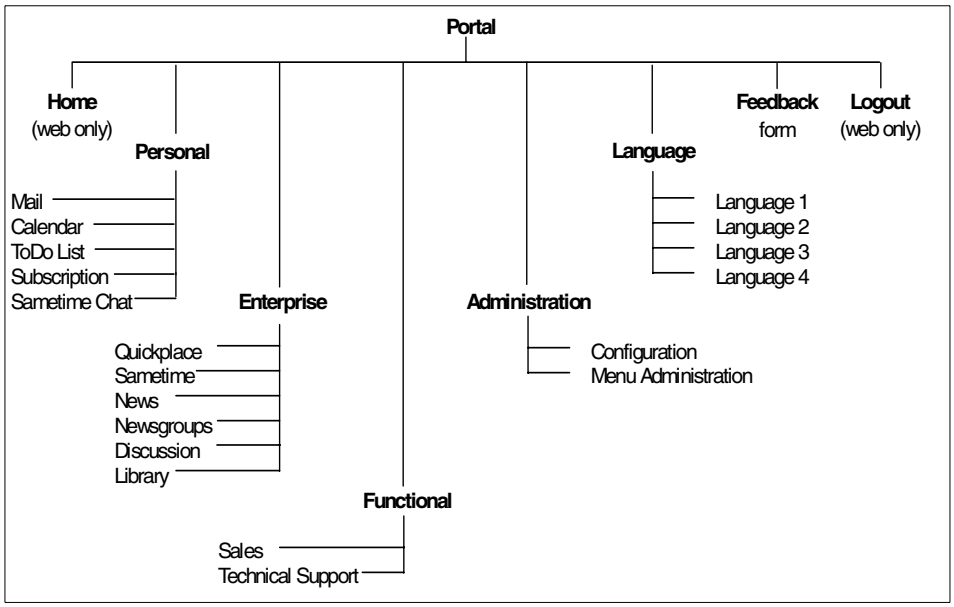


Figure 3. Our portal structure

We structured our portal into five basic levels of content, illustrated in Figure 3, as follows:

- Personal content, containing:
  - Personal mail
  - Personal calendar
  - Personal to do list
  - Personal subscriptions
  - Sametime chat
- Enterprise content, containing:
  - Quickplace, a team workspace on the web.
  - Sametime, a community of users who can collaborate in real-time online meetings
  - Enterprise news
  - Enterprise newsgroups
  - Enterprise discussions

- Enterprise document library

- Functional content, containing the following diverse functional material:
  - Sales applications
  - Technical support applications
- Administration functions that include:
  - Configuration functions
  - Menu administration
- Language selections for selecting a different language

We also added the following functions to the kernel portal:

- A link to the home page for quick access back to the main page (only in the browser environment)
- A feedback form for users to provide us with some feedback
- A logout button to allow users to log out of the portal (only in the browser environment)

3.3 The portal database

The first thing we do is create a database to support our portal application. The database will contain the portal's kernel structure. This will allow us to navigate through the central structure and access each application in our portal.

To create the portal database, follow these steps:

1. Open the Domino designer client. Select **File** -> **Database** -> **New**.
2. In the New Database dialog box:
   a. Leave the Server field set to "Local" during development. You can replicate this application to a server later.
   b. In the Title field, type a name for your portal.
   c. In the File Name field, type a file name for your portal.
   d. Leave the Template set to "Blank" and click OK to create your portal.

3.3.1 Mapping the individualized levels in the application

Once we have built our database, we set up the ACL content with users, groups, access levels, and roles. In order to apply the access control to the different applications contained in the portal, we did a mapping between the

individualized levels of content and the user roles that we included in the portal. Refer to 4.1, “Personalization with user roles” on page 55. The following table shows the relationships between access and roles.

Content	Application	Access
Personal	Personal	Username
Workgroup	Functional	Userrole/ Group
Enterprise	Enterprise	Default for Employees Group?
Extranet	Functional	Userrole/ Group
Internet	Functional (possibly separated)	Anonymous

In the application, we did not map the content to the application elements one on one. For the extranet content, we used the functional application elements. The same applies to the workgroup content and the Internet content.

3.3.2 Using access control for content mapping

3.3.2.1 Personal content

We automatically take control of the personal information for an individual user based on the particular user name, so it is impossible for one user to see the information of another. We use the @UserName function, which identifies the personal information for each individual user in the portal.

3.3.2.2 Workgroup content

We control the access related to an individual’s workgroup information based on the assigned role for the specific user, so a specific user only has access to the functional portals that match with their assigned roles. We use the @UserRoles function to identify the user role and orient the user to his corresponding functional portal.

3.3.2.3 Enterprise content

We opened this access to all the internal users of the company, and at the same time we excluded the external users of the company. We based this on

the role assigned to the individual user. We used the @UserRoles function to make this differentiation.

3.3.2.4 Extranet content

We opened this access to special external users we have included in our address book. We based this on the role assigned to the individual user and we used the @UserRoles function to make this distinction.

3.3.2.5 Internet content

We integrated external users not included into our address book based on anonymous access in the ACL of the portal database.

3.3.3 ACL user roles

We defined the following roles at the portal application:

- [PortalAdmin]
- [Sales]
- [TechSupport]
- [InternalUser]
- [ExternalUser]
- [Developer]

3.3.3.1 [PortalAdmin]

This role is assigned to users who are in charge of the portal administration. They must be able to configure the parameters needed for the portal application. We consider this the most powerful user role in the portal application.

3.3.3.2 [Sales]

Users with this role are able to get access to the sales portal page. The sales department is one of the special workgroups we included in our portal.

3.3.3.3 [TechSupport]

The users of this role are able to access the technical support page. The technical support department is one of the special workgroups we included in the portal.

3.3.3.4 [InternalUser]

This role identifies the users corresponding to employees of the company.

3.3.3.5 [ExternalUser]

This role identifies the users corresponding to people outside of the company.

3.3.3.6 [Developer]

This role is assigned to members of the developer team. It is very useful when we are required to maintain the portal and provide technical support for the application.

3.4 Designing the main portal page

The root element of our portal is the main portal page. Using this element, we are able to navigate across the content of the portal. This main page is divided into two sections, menu and body. The menu section contains options to navigate across all the content options. The body section shows the currently selected content page.

Figure 4. Main portal page layout structure

We created this structure using a Framesets design element, which we split into two frames. The left frame is named “MainMenu” and it contains an outline design element named “PortalMainMenu” that supports our choice menu. We included links to the element to get access to our personal page, enterprise page, functional page, and portal administration section.

Follow these steps to create the frameset:

1. Open Designer. On the design pane, click the Framesets design bookmark.
2. Click the New Frameset design action button. The Create New Frameset appears.
3. At the “Number of frames” drop down box, select two frames. Next to Arrangement click the first arrangement for the frames.
4. Click OK. The frameset with the chosen layout appears.
5. Open the Frameset Properties box and at the Basics tab:
   a. Name the frameset “Main Portal”.
   b. Add a comment. The comment appears with the initial list of framesets in the Work pane. Type “Main portal page (Principal frameset)“.
6. Click in the left frame and make sure you are at the Frameset Properties box.
7. At the Basics tab name the frame “MainMenu”.
8. Click on the right frame.
9. At the Basics tab name the frame “MainBody”.

3.4.1 Designing the portal menu structure

Once we have created our frameset structure, we can create the navigational mechanism that will be contained in the MainMenu frame. We required one outline design element that contains all the menu choices according to the user’s profile.

First we created an Outline design element named “PortalMainMenu” (alias “MainMenu”), then we added ten main entries.

- The first entry is only available in the browser environment and contains a link to the home page of the portal. The entry for the home page is named “Home” alias “Home”. At the “Hide outline entry from” field, we selected the “Notes R4.6 or later” checkbox. This will assure that the Home element only displays in the browser environment. We assigned the URL type to the element and pointed it to our server’s home URL.
- The next two entries are oriented to display the personal portal page, one for the Notes client and another for the Web browser.
  - The entry for the Notes client was named “Personal” alias “PersonalNotes”. At the “Hide outline entry from” field we selected the

"Web Browsers" checkbox. This will assure that the element only displays in the Notes client environment.

- For the browser client we named the entry "Personal" alias "PersonalWeb" and at the "Hide outline entry from" field we selected the "Notes R4.6 or later" checkbox to assure that the portal page will only be displayed in the Web browser environment.

- The fourth entry is oriented to provide access to the enterprise portal page, which is available to all the different kinds of users we'll have in our portal application. The entry was named "Enterprise" alias "Enterprise".
- The fifth entry is oriented to display the different functional portals the user is able to get access to. It is named "Functional" alias "Functional", and it is integrated for two indent entries, one for our sales portal page and another for our technical support portal page.
  - The first indent entry is named "Sales" alias "Sales". At the "Entry Hide When" field, select "Hide outline entry if formula is true" and write the formula @IsNotMember("[Sales]";@UserRoles). This ensures that the sales portal page will be displayed only if the user is assigned the "[Sales]" role.
  - The other indent entry is named "TechSupport" alias "TechSupport". At the "Entry Hide When" field, select "Hide outline entry if formula is true" and write the formula:@IsNotMember("[TechSupport]";@UserRoles). This ensures that the technical support portal page will be displayed only if the user is assigned the "[TechSupport]" role.

  One user could have access to many different functional portal pages, according to the access roles they were assigned.
- The sixth choice provides access to the configuration elements of our portal application. Only users with the "[PortalAdmin]" role assigned can get access to this configuration elements.

  The entry is named "Administration" alias "administration". At the "Entry Hide When" field, select "Hide outline entry if formula is true" and write the formula: @IsNotMember("[PortalAdmon]";@UserRoles).

  We also included the indent entry "Configuration" alias "Configuration" as well as the indent entry "Menu Admin" alias "MenuAdmin". These entries will be described in Chapter 6, "Integrating personal and enterprise pages" on page 105.
- The next two entries were made to accommodate the language selections. We named them both "Language" with aliases "LanguageNotes" and "LanguageWeb" for the respective environments. Again we selected the correct checkbox for the element to display in either the Notes client

environment or the browser environment. The use of the language selection will be described in Chapter 4, “User personalization” on page 55.

- The ninth entry contains the Feedback form and was named “Feedback” with alias “Feedback”.
- Finally, we added the last element, called “Logout” with alias “Logout”. This element points to a URL, which must be your server’s URL followed by “?Logout”. Again the element is set to only display in the browser environment. This element will give a user the means of logging out of the portal.

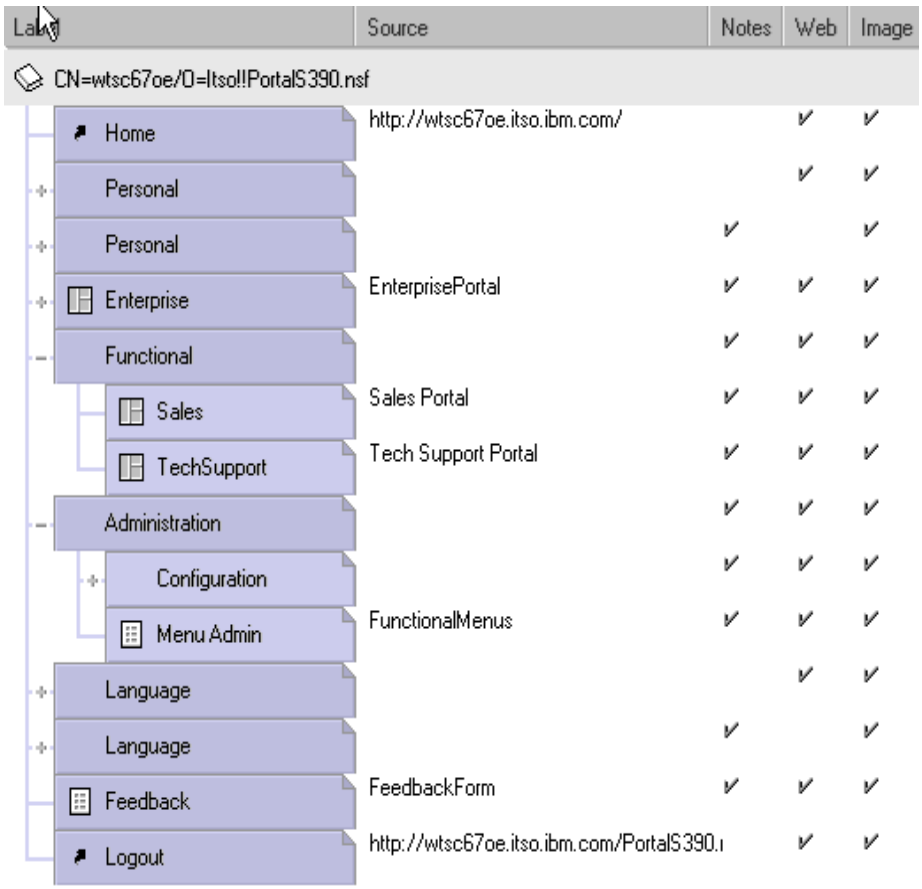


Figure 5. “Main Menu” outline design element

After we have completed our outline design element, we created the form in which we embedded this outline element.

The form was named “MainMenu”. We created a table with two rows and one column. We put the logo of the company in the first row. In the next row we created a field to show the name of the current user, with the following properties:

```
Field name UserName, Type = Text/Computed for display,
Value=@Name([CN];@UserName)
```

Figure 6 shows the form after the outline was added.

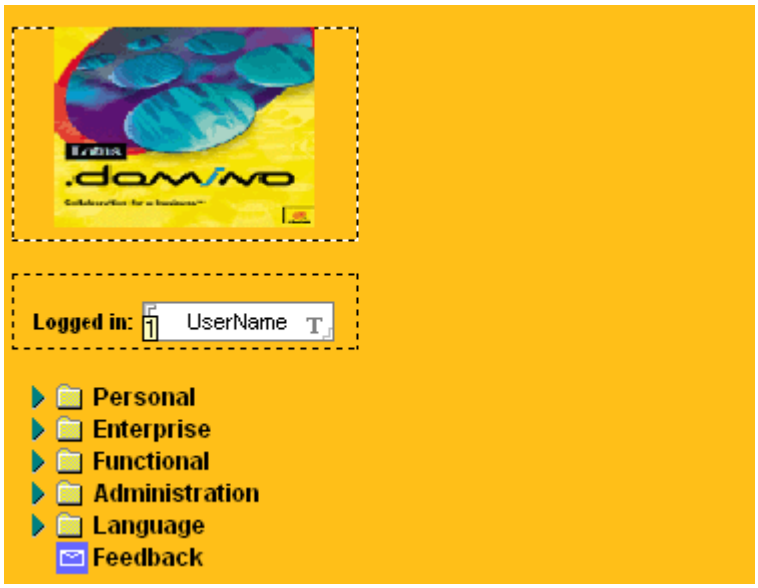


Figure 6. “Main Menu” form

3.4.2 Designing the enterprise portal

After completing the menu side of the frameset, we created a new frameset imbedded in the body side of the main frameset and added it to the Body frame. We made the enterprise portal the entry point to the portal and showed the enterprise portal pages in the body frame when a user accesses the portal.

We divided the enterprise portal pages into two frames. The top frame will contain a search option that is already available in Lotus Domino. The bottom frame will contain the enterprise portal pages, with an initial welcome page.

To create the enterprise portal pages, follow these steps:

1. Open Designer and on the design pane click the Framesets design bookmark.

2. Click the New Frameset design action button. The Create New Frameset screen appears.
3. At the “Number of frames” dropdown box, select two frames. Next to Arrangement click the last arrangement for the frames.
4. Click OK. The frameset with the chosen layout appears.
5. Open the Frameset Properties box and at the Basics tab:
   a. Name the frameset “EnterprisePortal“
   b. Add a comment. The comment appears with the initial list of framesets in the Work pane. Type “This is the enterprise portal page“.

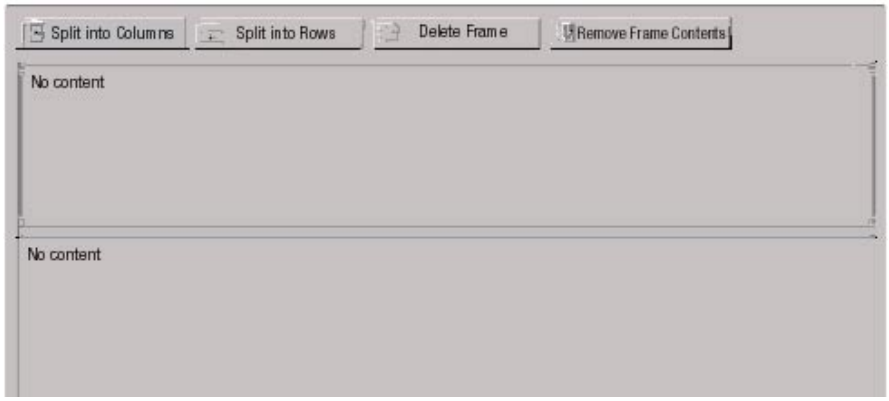


Figure 7. Enterprise portal page layout structure

In Chapters 6 and 7 we will complete the content for these frames. For the moment, we have completed our portal’s main page structure.

3.5 Designing the personal portal pages

Users can get access to their personal content through the selection of the personal portal entries on the menu. In this section we included five of the most useful applications to show personal information related to the user. The applications are mail, calendar, contacts, tasks to do, subscriptions, and Sametime chat.

All the function required is already included in the Lotus Notes infrastructure, so it was minimal effort to integrate all these elements into the personal page. We first used a frameset design element and included the mail, calendar, tasks to do, and subscriptions into the frameset. This gives a user one page with all his personal applications included in the frame. As we began to use

these applications, we came to the conclusion that a single page confuses the user and we ended up dragging the frames bigger and smaller all the time, depending on which application we wanted to use.

To eliminate this inconvenience, we added all the applications as menu entries into the menu outline element and created a separate page for each application. We also changed the settings, to open each application into its own window. This allowed us to keep, for instance, our mail open all the time, while working in other applications in the portal.

To build the Notes client solution, we began with the instructions in *Building a Portal with Lotus Domino R5*, REDP0019. For the Web browser client solution, we made some modifications, including adding HTML and LotusScript code, in order to meet our goal of keeping the same functionality and look in both environments.

In Chapter 4, “User personalization” on page 55, we’ll describe the content of these pages.

3.6 Designing the functional portal pages

Our portal included two examples of functional portal pages:

- The sales portal
- The technical support portal

Inside the portals, we included some of the most useful collaborative applications that are provided with standard templates by the Lotus Domino infrastructure; for example, the teamroom application.

In the sales portal we included an invoice application to show how Lotus Domino can connect to DB2 on a S/390 system using the LCLSX programming connection. We also included a customer database application to show how Lotus Domino can connect to DB2 on a S/390 system using the DECS connector.

We included a parts ordering application in the technical support portal, which uses MQseries and CICS to execute transactions on a S/390 system.

Follow the next steps to create a functional portal pages:

1. Open Designer and on the design pane click the Framesets design bookmark.

2. Click on the New Frameset design action button. The Create New Frameset window appears.
3. At the “Number of frames” dropdown box, select two frames. Next to Arrangement click the first arrangement for the frames.
4. Click OK. The frameset with the chosen layout appears.
5. Open the Frameset Properties box and at the Basics tab:
   a. Name the frameset “Sales“.
   b. Add a comment. The comment appears with the initial list of framesets in the Work pane. Type “This is the sales functional portal page“.
6. Click on frame 1 and at the Basics tab name the frame “SalesMenu”.
7. Click on frame 2 and at the Basics tab name the frame “SalesBody”.

Repeat the same steps to create the corresponding functional page for the technical support workgroup. The same structure was used for this.

In Chapters 6 and 7 we’ll complete the content for these frames. For the moment we have completed our functional portal page’s structure for both Notes and browser clients.

3.7 Designing the administration section

We included an administration section in the portal for users with administration access to use. We added a menu entry for a configuration area and a menu administration area.

These applications will be described in more detail in Chapters 6 and 7.

3.8 Extranet and Internet applications

We decided to concentrate on an intranet portal, rather than an extranet or Internet portal. To cater for extranet applications we created roles with access that can be controlled using the roles “InternalUser” and “ExternalUser”, to make a distinction between internal users and external users of the company.

These roles can be used for access, but we did not use them as we focused on the intranet applications.

3.9 Special considerations

When designing a portal that can be used both in the Notes client and browser environment, some special considerations must be taken. Things to consider are:

- Security aspects
- Network traffic
- The usage of scripts and formulas
- The layout of the user interface
- Methods used to open objects
- Domino for S/390 and Web server integration
- Domino and WebSphere
- Methods to open the portal

3.9.1 Security aspects

Portal security is covered in detail in Chapter 5, “Portal security” on page 77. However, some basic things to keep in mind are the way that a Web browser accesses a page and the way that a Notes client accesses a database.

It is important to provide a good login interface for the browser environment with a logout button and session time-out limits. Remember, the browser environment can theoretically access pages from anywhere in the world under various conditions. Someone may access the pages from an Internet cafe or a shared computer. The Notes client is a much more controlled environment and security is built into the Notes architecture.

Another aspect that must be considered carefully is the kind of network that is used for the shipment of the data. At least server-side SSL should be implemented if an unsecure Internet network is used in the browser environment, depending on the sensitivity of the data. Notes clients traditionally access data in a much more secure network environment.

3.9.2 Network traffic

Networks are one of the most important issues when it comes to performance of a client server type of application. A network can seriously damage the performance of a normally good application. Notes clients traditionally operate in a more secure and well-managed network, while a browser client can operate in any kind of network.

You should consider the amount of data that is shipped over the network as well as what kind of data is shipped. In the browser environment a lot of network traffic can be eliminated if, for instance, client-side javascript is used for validating input fields before a form is submitted.

Java applets and images are two other network-intensive elements that should be carefully considered when designing the portal. It could, however, be better to download a Java applet once and let it do a lot of work on the client, rather than sending data from the client to the server each time.

3.9.3 The usage of scripts and formulas

Figure 8 shows the programmatic access to the Notes UI.

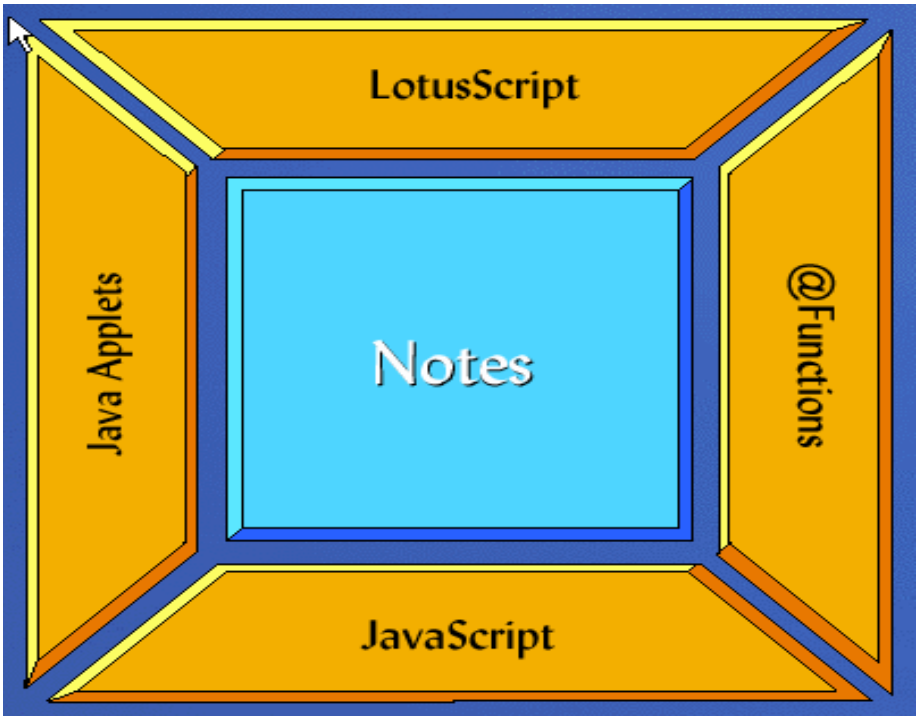


Figure 8. Programmatic access to the Notes UI

We followed the guidelines that are documented in the Domino 5 Designer help file when we used scripts, formulas, and applets.

Before you write code, see if a simple action will do the task. You can design some objects with simple actions in forms or views that do not require programming.

Where you have a choice among programming interfaces, consider these guidelines:

- Formulas are expressions that have program-like attributes. For example, you can assign values to variables and use a limited control logic. The formula language interface to Domino Designer is through calls to @functions and @commands.

  In general, formulas are best used for working within the object that the user is currently processing, for example, to return a default value to a field or determine selection criteria for a view. Additionally, formulas provide better performance in some situations and may be more convenient for simple applications.

- JavaScript is a cross-platform, object-oriented scripting language. Header scripts may be written in the Programmer's pane by selecting JS Header from the Objects tab and typing your script in the Script Area. Scripts may also be attached to specific events such as onClick, or to objects such as buttons. JavaScript may not be written in an agent. Domino oversees the compilation and loading of user scripts, but does not store JavaScript in compiled form.

  JavaScript is best used for Web applications, or when a single application will be used in both the Notes and Web environments.

- LotusScript is a full object-oriented programming language. Its interface to Domino is through predefined classes. Domino oversees the compilation and loading of user code, and automatically includes the Domino class definitions.

  LotusScript is best used where the programming logic is not simple. LotusScript excels in accessing stored databases (back-end). LotusScript provides some capabilities that formulas do not, such as the ability to manipulate a database access control list (ACL). The UI (front-end) capabilities of LotusScript are limited.

- Java is a full object-oriented programming language. Its interface to Domino is through predefined classes. It is comparable to LotusScript in agents but cannot be attached to events in the Domino UI. Domino oversees the compilation and loading of user code for agents; the code can be written natively or imported.

  Java can be used in agents. Java applications and applets, written and compiled outside of Domino, can access Domino through the class interface.

3.9.4 The layout of the user interface

Special care should be taken with the layout and design of the user interface. Things to consider that may differ in a Notes client environment and the browser environment are:

- The font size of the text
- Scrolling of windows, frames, and objects
- Validation of input fields
- Cursor positioning with the initial load of a page or form
- Tab orders
- Button sizes
- Navigation methods
- Performance

All of these elements looked fine while we developed in, for instance, the Notes client environment, but had to be re-visited when we accessed the pages in the browser environment, or vice-versa. Small differences can be very irritating for a user if the application does not work as intended. We also found that these items could take a lot of time to correct. Sometimes we had to compromise the look and feel of either the Notes client environment or the browser environment to get the best effect.

Our advice is to leave enough time in a project schedule to thoroughly explore and test these aspects of the user interface. Remember, an application can be functionally perfect, but the users have to work with the user interface. In the end it is they who decide whether an application is good or not, and their judgement can hinge much more on the user interface than on the functionality.

Note

Remember that you can use Javascript and cascading stylesheets, especially in the browser environment.

3.9.5 Methods used to open objects

It is important to look at the ways that you open pages, forms, views, and links. Sometimes it is necessary to use two different methods when working in the Notes client and browser environment and to hide the method, depending on the environment.

We found that it is very easy to first implement a function in the browser environment and then to use the same URL link for the Notes client. But, then the link opens in the Notes client browser and not in the Notes client itself. This has a performance penalty as well as a security aspect involved. It is preferable to open a named element or action when working in the Notes client environment, rather than a URL link. Also, try to find ways to use one method that caters to both environments. This makes the maintenance of the portal much easier later on.

3.9.6 Domino for S/390 and Web server integration

Our portal was designed in the S/390 environment, so we don't include much detail about Domino for S/390 and Web server integration in this redbook. The concept is described in detail in *Domino for S/390 and Web Server Integration*, SG24-5437.

The Domino for IBM HTTP Server is an IBM HTTP Server API (GWAPI) program that connects Domino 5.0.1 to the IBM HTTP Server. It is usually referred to as the "Web connector." The Web connector allows you to access Domino databases using a Web browser that is connected to a Web server on OS/390 Web (IBM HTTP Server for OS/390). The Web connector runs as a GWAPI program in the IBM HTTP server tasks and accesses Domino databases using Domino APIs. Based on the GWAPI construct, most of the workload is shifted from the Domino server to the IBM HTTP server.

Domino R5 for OS/390 still provides an internal Web server in addition to this new feature. The preferred option to access Domino is to use the Notes client software; it is still the most efficient way to access Domino applications.

The Web server connectivity option is preferred for thin clients. Now, since there are two different methods of accessing Domino databases using the Web interface, your question will very likely be, which interface should you choose?

You probably want to use the Domino for IBM HTTP Server interface in the following cases:

- If you want to use Domino data or Domino access as part of a Web site that will also contain other fancy Web elements such as animation or multimedia. In this case you probably want to serve pictures, frames, multimedia and other components directly from the IBM HTTP Server without having Domino involved, and display the Domino application as part of a browser frame. This can be important in terms of performance since the IBM HTTP Server has several performance features like the FRCA cache. When it comes to static Web serving, such as accessing

pictures and other static objects, then there is an advantage in using the OS/390 Web server.

- If you want to run servlets and Java Server pages through the IBM WebSphere Application server rather than through the Domino Servlet Manager.
- If you want to be more flexible and have more influence on workload balancing by using the scalable Web server or by splitting the Domino application so it can be accessed through different Web servers.
- If you want to use OS/390 user authentication.

Reasons *not* to use the Domino for IBM HTTP server are:

- If you only need to Web-enable the Domino database access and you are not experienced in the OS/390 UNIX environment, especially not in the OS/390 Web server-OS/390 WebSphere Application Server environment, you will probably just set up the internal Domino Web interface. This option also has the advantage of being consistent across the platforms that support Domino.
- If you are mostly accessing Domino using the client software and the few Web accesses would look too different from the client interface appearance.
- If you want to use the Domino Internet Cluster Manager.
- If you want to use the Domino Web server API (DSAPI) filters.

You are not limited to choosing just one Web interface, either the internal Domino Web interface or the Web connector. You may in fact combine both methods if you need to use functions provided by one or the other.

3.9.7 Domino and WebSphere

WebSphere's support for Domino allows Java servlet applications to be written to run in either environment and allows the same base Java servlet code to execute in both products. Applications can also call back and forth between the WebSphere and Domino environments through the use of Java. In addition, the capabilities of Domino and WAS can be combined in a server farm, where multiple Web servers work together transparently to fulfill the needs of a single Web-based solution.

Together, WAS and Domino offer a comprehensive range of Web application server environments that support business applications—from simple Web publishing through enterprise-scale transaction processing and collaborative business solutions.

3.9.8 Methods to open the portal

As described earlier, one of the advantages of having a portal is to lessen the time-consuming process of users trying to find information. It is all well and good to have a portal, but if your users have to search for the portal it immediately becomes useless and almost obsolete.

So, how can you help users to find a portal? We will describe two ways to do this, one for a browser and one for a Notes client. Basically you want users to be able to access everything from their portal. That means that when users launch the Notes client the portal should be launched automatically. The same goes for a browser. But first, you have to ensure that when the portal is launched, it will open up in the correct frameset.

To ensure that the portal database opens to the correct frameset every time, you have to set the launch settings on the database properties page as shown in Figure 9. Remember to set it for both the Notes and the browser client.

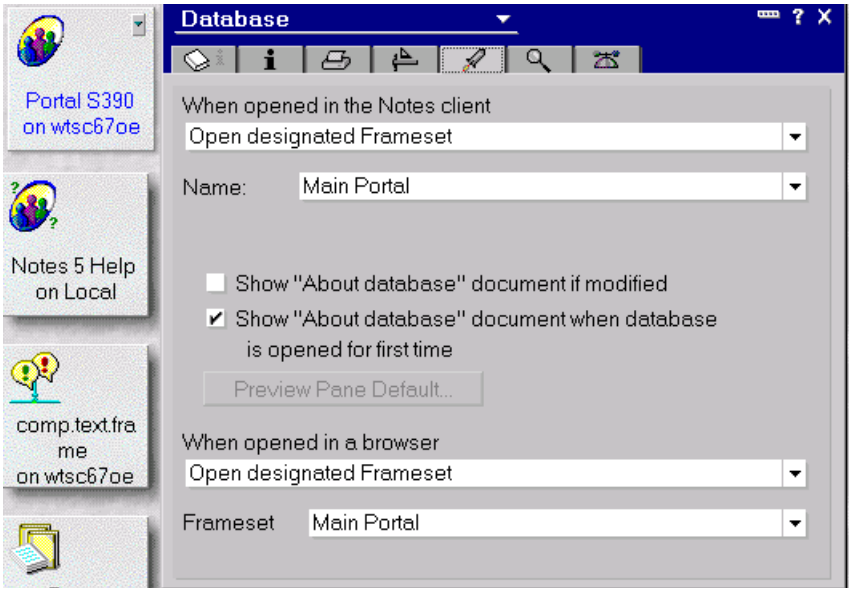


Figure 9. Database settings to launch the portal

3.9.8.1 Opening the portal from the Notes client

Setting a database to automatically open when you launch Notes is done from the Notes client. It involves two steps:

1. Bookmark a database.

2. Set the database as your Notes “homepage.”

Once you have bookmarked a database, document, or a Web page, you can set that bookmark to open every time you start Notes. In previous versions of Notes, you could choose a startup database. In Notes Release 5, you select a bookmark as your home page, and that database, document, or Web page will open as your default view when you start Notes. You can redo this as often as you like. You can also reset the Welcome Page to act as your home page.

Bookmark a database
To bookmark a database:

1. Choose **File -> Database -> Open**.
2. Enter the name of the server in the Server box to see a list of available databases.
3. Select the portal database from the list of available databases on the server, and click Bookmark.
4. Select the name of the Bookmark folder to which you would like to add the bookmark.

Set the Bookmark as your homepage
To set a bookmark as your home page:

1. Right-click on any bookmark.
2. Select the option, “Set Bookmark as Home Page.”
3. In the “Set Home Page” dialog box, click OK.

If the desired database does not have a bookmark, drag the open database's task button to the bookmark bar.

To undo setting a bookmark as your home page
You can reset the default Welcome Page to act as your Home Page.

1. Open your Favorite Bookmarks folder on the bookmark bar.
2. Right click on the Welcome bookmark.
3. Select the option “Set Bookmark as Home Page.”
4. In the “Set Home Page” dialog box, click OK.

3.9.8.2 Opening the portal from a browser
We all know that we can set a homepage for a browser. Ideally, one should set up users’ browser home pages so that the portal is the first page they see

when opening the browser. This is, however, a function that must be performed on the client and users just love to play around with browser settings. You cannot force a user to have the portal as his or her home page, but you can enforce certain settings on the Domino server serving the portal pages (refer to Figure 10 on page 52) using the following steps:

1. On the server document, go to the Internet Protocols, HTTP tab.
2. In the Basics section, ensure that you have the fully qualified internet host name as defined in the DNS.
3. In the Mapping section Home URL field, enter the path to your portal database. This is the full path relative to your Notesdata directory. For example, if your portal is in a subdirectory under Notesdata, your URL would look something like this:

   ```
   dir/portaldb.nsf?Open
   ```

To ensure that the database opens correctly on the browser, you have to set the database launch properties to open the designated frameset, as shown in Figure 10.

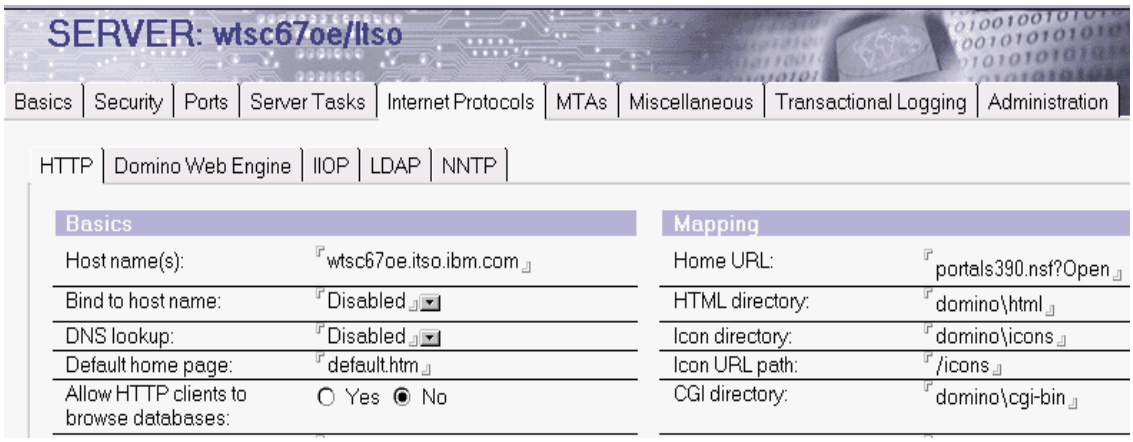


Figure 10. Server setting to automatically launch the portal from a browser

In addition to the Home URL, you can also set the “Allow HTTP clients to browse databases” field in the Basic section to `No`.

This setting also brings you more security to the Domino server because users are not allowed to browse databases on the server. The only way they could get to other databases from a browser is if they know the exact filename of the database and how to open it from a browser. In this case, you still have the security of the database ACL. To ensure security for your

Domino databases, always ensure that your ACLs are set up properly. You must *at least* ensure that your system databases (admin.nsf, names.nsf, log.nsf) are properly secured by setting default and anonymous access to “None.”

Chapter 4. User personalization

One of the most important features in a portal is personalization. This is a fundamental difference between a portal and a framed intranet site. Portals that target information to specific users are a great boost to knowledge management initiatives in an organization.

Examples of portal personalization include allowing the users to see their own mail, calendar and scheduling information.

A way to further personalize the portal is to open the portal at the user's particular workgroup portal home page. When the user opens the portal, it checks their identity and opens the appropriate frameset for them to view. One way to achieve this is by defining user roles in the portal access control list and tying each individual user to a specified role.

Another important aspect of personalization is the capability to choose the language of your choice.

4.1 Personalization with user roles

Every database includes an access control list (ACL), which Notes uses to determine the level of database access that users and servers have. As an additional level of security, you can also create one or more roles and assign each user to a role. For more information about the database ACL and roles, refer to 5.3, "Database access control" on page 98.

In the portal database, the role assigned to each user forms the basis on which specific framesets are opened for them.

To enable personalization using roles, you can, for example, embed formulas in the design to prevent users who don't have these roles from accessing the corresponding element of design. In the portal database, we used this function to tailor entries on the outline menu for members of the different groups. For example, the outline entry in Figure 11 on page 56 would not be visible to users that do not have the [Sales] role assigned to them.

© Copyright IBM Corp. 2001

.

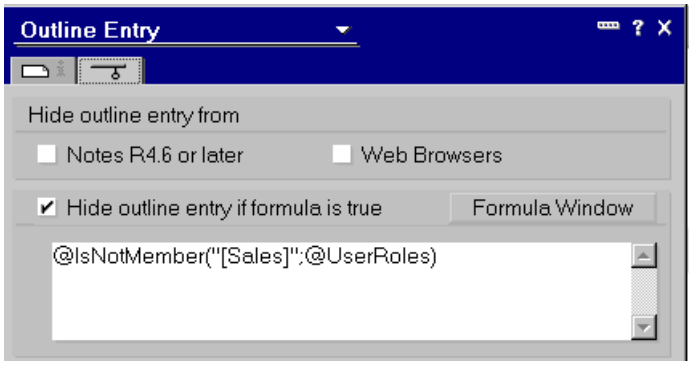


Figure 11. Outline entry

For more details on how we designed the portal access and roles, refer to 3.3.1, “Mapping the individualized levels in the application” on page 33.

4.1.1 How to set up roles

After registering sample users, we assigned them to groups, added the groups to the portal database ACL, and assigned each group to one or more roles.

4.1.1.1 Register users

We registered the users as Notes mail users, with an Internet password. They can access the portal database with either a Notes client or an Internet browser.

In this sample portal application, four people are registered, one for each group, and one not belonging to the company.

```
Tom Arrell belongs to the PortalAdmin group.
Allan Smith belongs to the PortalSales group.
Ken Tucky belongs to the PortalTech group.
Alison Pier doesn't belong to a group.
```

4.1.1.2 Create groups and assign roles

Users are registered as belonging to one or more groups, according to the function they belong to. To manage access to certain design elements, we created three groups and assigned each of these groups a role in the portal database ACL:

- **PortalAdmin**

  This group contains the portal database administrators.

Members of the group need, at least, to have the [PortalAdmin] role assigned. In the portal database ACL, we set it with Manager access and all the roles enabled, as shown in Figure 12.

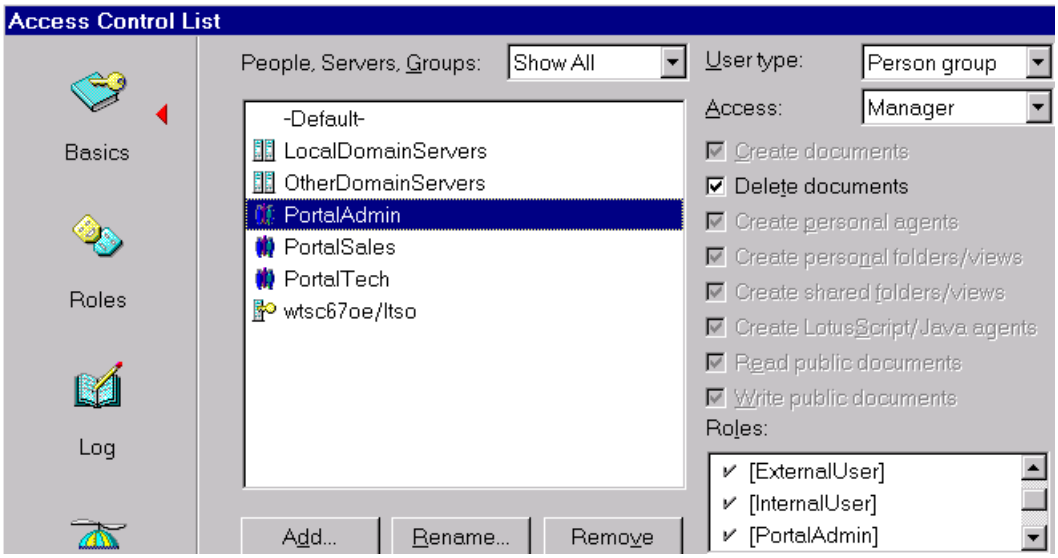


Figure 12. Defining access for the Portal group

- **PortalSales**

  This group is defined for the Sales Functional Portal and we assign them the [Sales] role. Users with this role can access the Sales portal page; see Figure 13 on page 58.

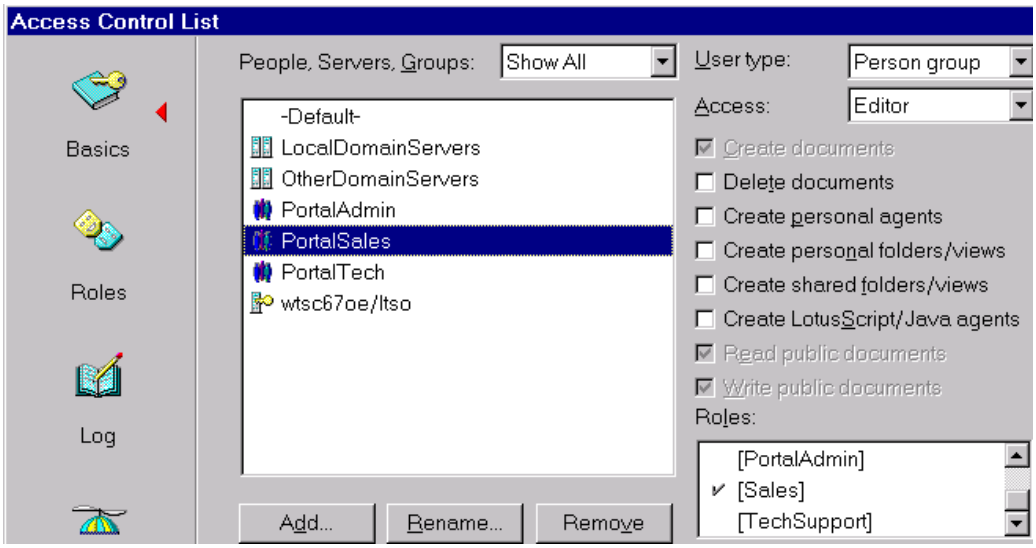


Figure 13. Defining access for the Sales group

- **PortalTech**

  This group is defined for the Technical Support Functional Portal and we assign them the [TechSupport] role. Users with this role can access the Technical support page; see Figure 14.

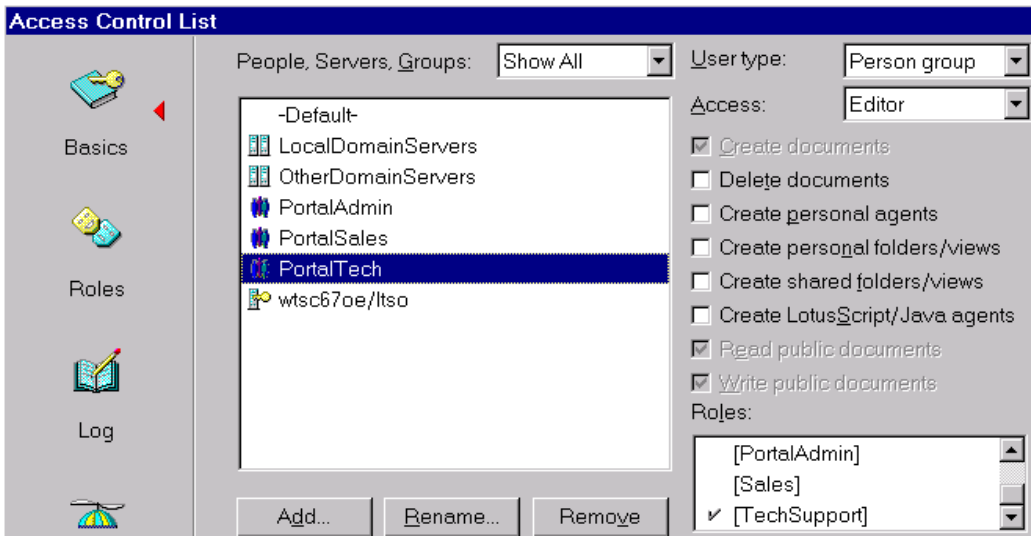


Figure 14. Defining access for the Tech Support group

4.1.1.3 Personalization result

Each person accessing the Web site will see a personalized portal desktop, according to his own function.

For example, Allan Smith, one of the Sales entity employees, gets the portal main page shown inFigure 15 when he logs in. Neither the TechSupport entry nor the portal administration entry are shown.

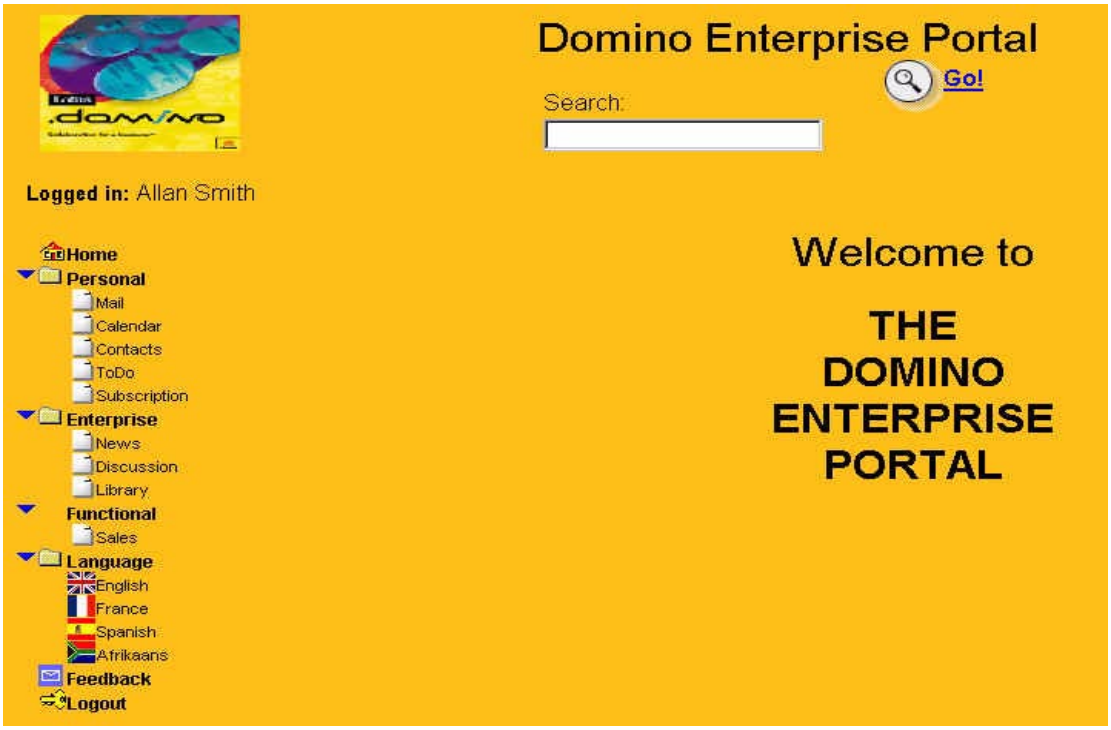


Figure 15. Portal homepage for Allan Smith

4.2 Making a multi-lingual portal

Multi-lingual (globalized) portals are capable of presenting information in more than one language. By 2002, according to IDC, 60% of business-to-business companies will be building globalized Web sites designed to reach audiences in many countries and cultures. In 1999, 37% of B-to-B sites were globalized. By 2004, the level of globalization is expected to level off at around 80%, and 50% of online sales will occur outside the United States. Firms that cling to English-only sites will cede their share of this

business to more globally-minded competitors. Globalizing your Web site is no longer a luxury if you would like to do business in other countries.

4.2.1 Tools for Web site translation

Here are several tools you can use to provide a multi-lingual portal.

Domino Global Workbench (DGW), is a fully integrated component of the Domino Designer R5 development environment. DGW enables Web application developers to quickly and cost-effectively deliver multilingual Web and intranet applications.

Lotus Translation Component is a set of programming objects and software services that allows companies to rapidly integrate translation services capability into their Sametime and Domino Web applications.

The **IBM WebSphere Translation Server** provides a robust, scalable, enterprise-level, machine translation solution for WebSphere, Domino, and Sametime, utilizing IBM's industry-leading quality translation engines, the Lotus Translation Components, and the Lotus Translation Services for Sametime.

4.2.2 Using Domino Global Workbench

Localizing a database or a template means creating a new copy of this database, translated in another language. Our portal database was translated using the Domino Global Workbench.

Domino Global Workbench is a set of tools designed to manage the localization (translation) of the Domino databases—especially Web sites databases.

Domino Global Workbench analyses the design elements, translates the text in them, and modifies the layout. It is primarily concerned with translating the design, not the content of the database, but it includes some synchronization features to help with this.

An advantage in using DGW is that only one design exists for the whole language's databases.

Domino Global Workbench Release 5.0.5 works with Domino Designer 5.0 and above. It can also build Notes R4.6 unilingual databases. For specific requirements, refer to the DGW Release Notes.

4.2.3 Database translation process

1. Prepare the source database.

   If new, design the application with a standard naming convention. This means using aliases when possible, and also prefixing the design elements with significant values. It is important to know whether an element has to be translated or not before beginning the whole process. Refer to Step 1 in Figure 16, “Overview of the DGW localization process” on page 62.

2. Select or create a glossary.

   The glossary is created empty and is filled later in the process. Existing terminology can be imported. See Step 2 in Figure 16.

3. Create the Tagged database.

   This is the same database as the source. Each design element is given a unique ID called a *tag* that permits this element to be linked to the glossary.

4. Create the final multi-lingual database.

   Choose a language and translate the glossary. A pseudo translation step can be performed to test the impact of translation on the application behavior. Refer to Step 3 in Figure 16.

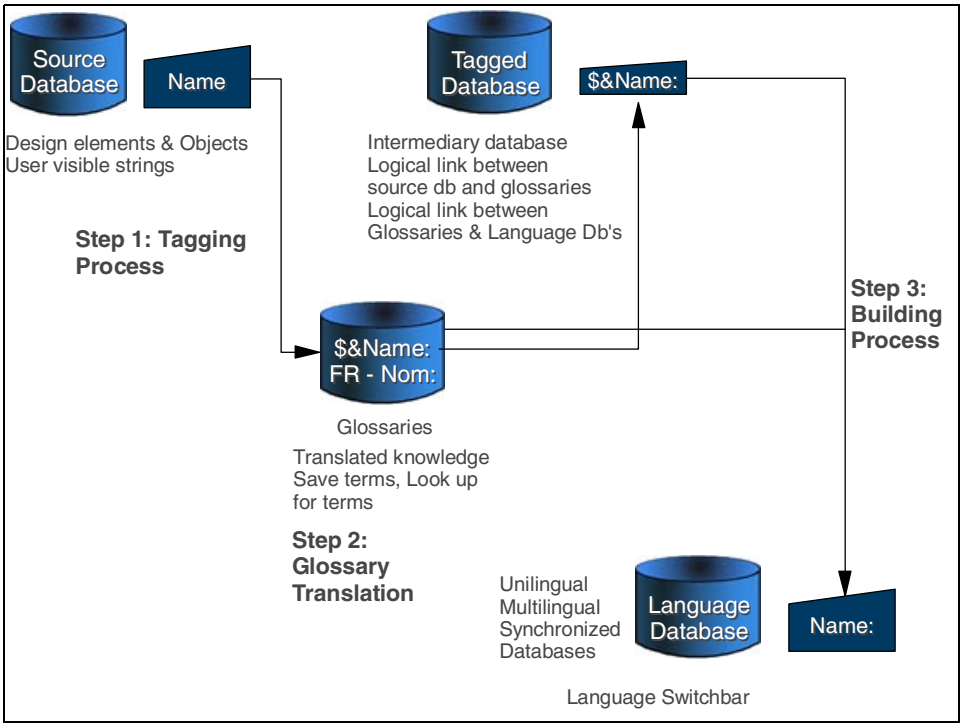


Figure 16. Overview of the DGW localization process

4.2.4 Multilingual database generation

Here are the steps to generate a multi-lingual database.

1. Install the Domino Global Workbench on your local workstation.

   If you receive a message that the Notes executable path is not found, exit the install program and add the lotus\notes path to your local environment settings.

2. Create a new project.

   From the Windows Taskbar, click **Start-> Programs -> Lotus Applications -> Domino Global Workbench**. You will see a panel like Figure 17 on page 63.

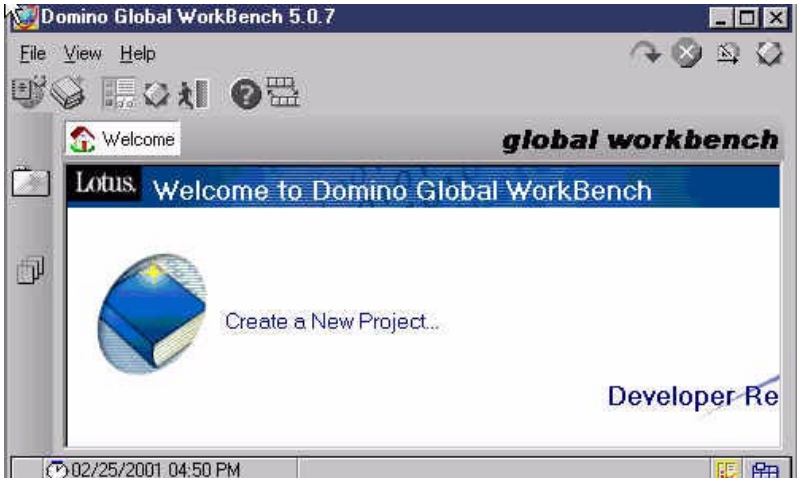


Figure 17. DGW welcome panel

On the “Welcome to Domino Global Workbench” panel, click **Create a new project** and enter the project name: `Portal S/390`.

3. Choose the database to be translated.

   Select **Click Here to Choose the source Database(s) to be Localized** and choose the portal database.

4. Create the Glossary database.

   Select **Click Here to Choose / Create the Glossary(ies) for Portal S390** and choose the portal database. See Figure 18. Click **Create** to create a new glossary.

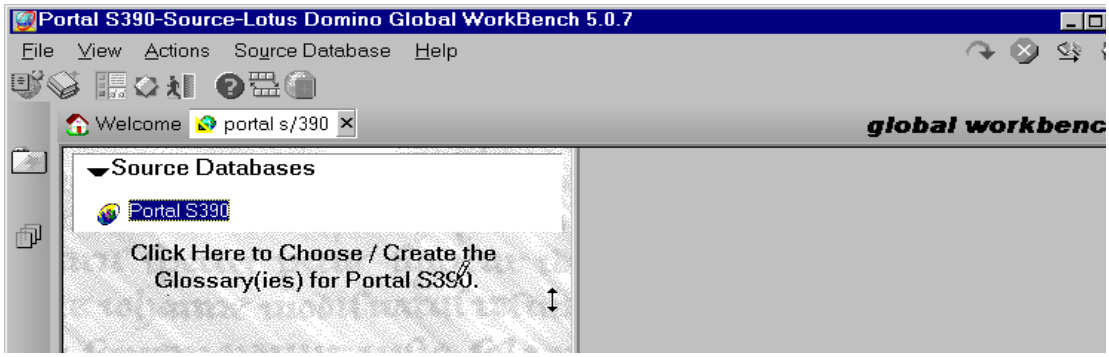

Portal S390-Source-Lotus Domino Global WorkBench 5.0.7
File View Actions Source Database Help
Welcome
portal s/390
global workbenc
Source Databases
Portal S390
Click Here to Choose / Create the Glossary(ies) for Portal S390.

Figure 18. Creating the glossary for portal S/390

Choose a server. In Figure 19 we chose **Local**, but it could be the Domino server on the host. Select the **DGW Glossary template** from the template dropdown list. From the Lotus Glossary Information panel, select first the reference language (English - United States) and then select the language(s) needed for the multi-language portal: Afrikaans, French, and Spanish.

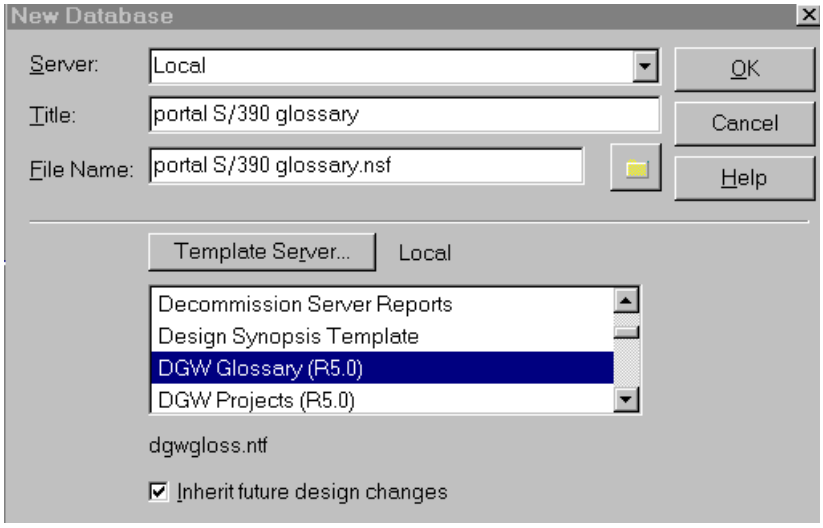


Figure 19. Choosing the template server

5. Create the Tagged Database.

   Select from the “name” dropdown list all the elements of design to be tagged. Click **Create Tagged Database**, as in Figure 20 on page 65.

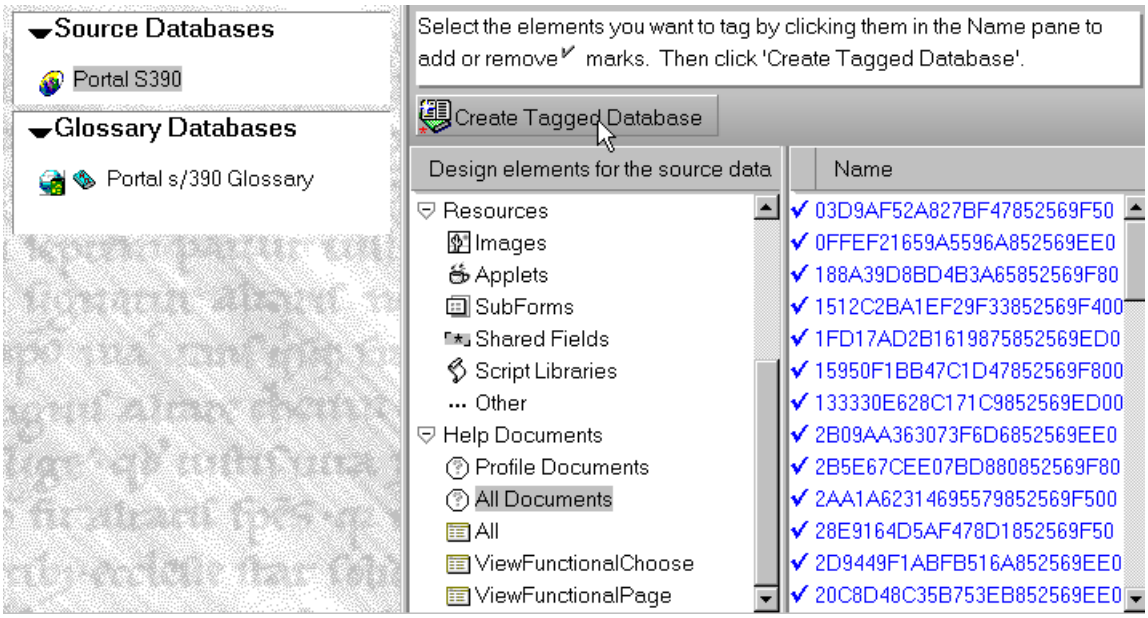


Figure 20. Creating a tagged database

Choose a server location (local or remote), then enter a name for the tagged database: Portaltg.nsf. The selected design elements are added in this new database as tags. This operation can take several minutes, depending on the number of design items that are to be tagged. Figure 21 on page 66 shows the items being tagged.

Note

The help design elements are not selected by default.

On one hand, it is better to add them to the selection, but on the other hand, their number must be kept as small as possible because they slow down the Workbench update procedure.

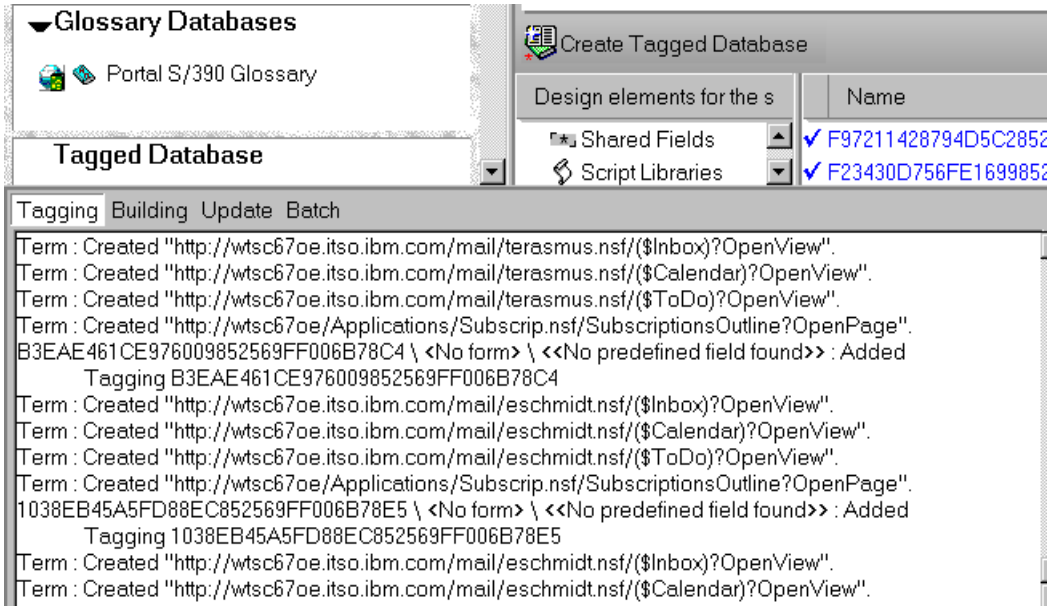


Figure 21. Items being tagged

6. Generate the language database(s).

   In the tagged database area, click the Language database dropdown list, and select **Add Language**, as shown in Figure 22.

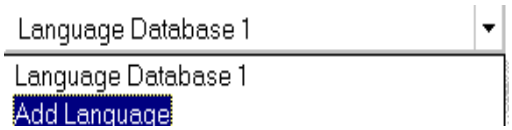


Figure 22. Choosing a language

   On the **Choose languages** panel, drag and drop a language entry, for example Spanish. See Figure 23 on page 67.

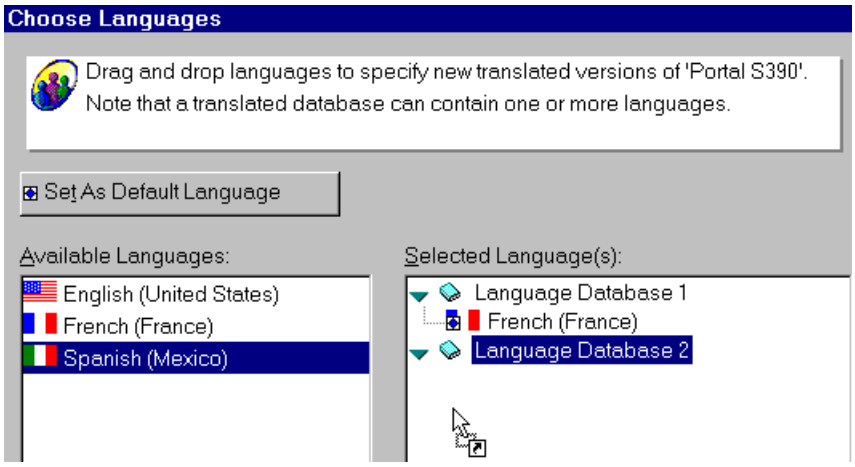

Choose Languages
Drag and drop languages to specify new translated versions of 'Portal S390'.
Note that a translated database can contain one or more languages.
Set As Default Language
Available Languages:
English (United States)
French (France)
Spanish (Mexico)
Selected Language(s):
Language Database 1
French (France)
Language Database 2

Figure 23. Selecting a language

When selected, click **OK**, then click **Build Language Database**, as shown in Figure 24.

Figure 24. Build language database button

On the **Create Language Database** panel, enter the server and the new language database name: Portal_fr.nsf. On the same panel, select the **Options** panel (Figure 25).

Figure 25. Options panel

Check **Copy ACL from source database**, if you want to keep the original database ACL (Figure 26).

Figure 26. Copy ACL from source database

The application now generates the Spanish language database. Other languages databases can be generated in the same way.

4.2.5 Reviewing the glossary

When you create a new glossary, as was the case for this portal database, you have to supply translated words, which have previously been identified as those that are displayed when opening the database.

In this portal example, we decided to translate the Domino Enterprise Portal homepage from the original English (US) language.

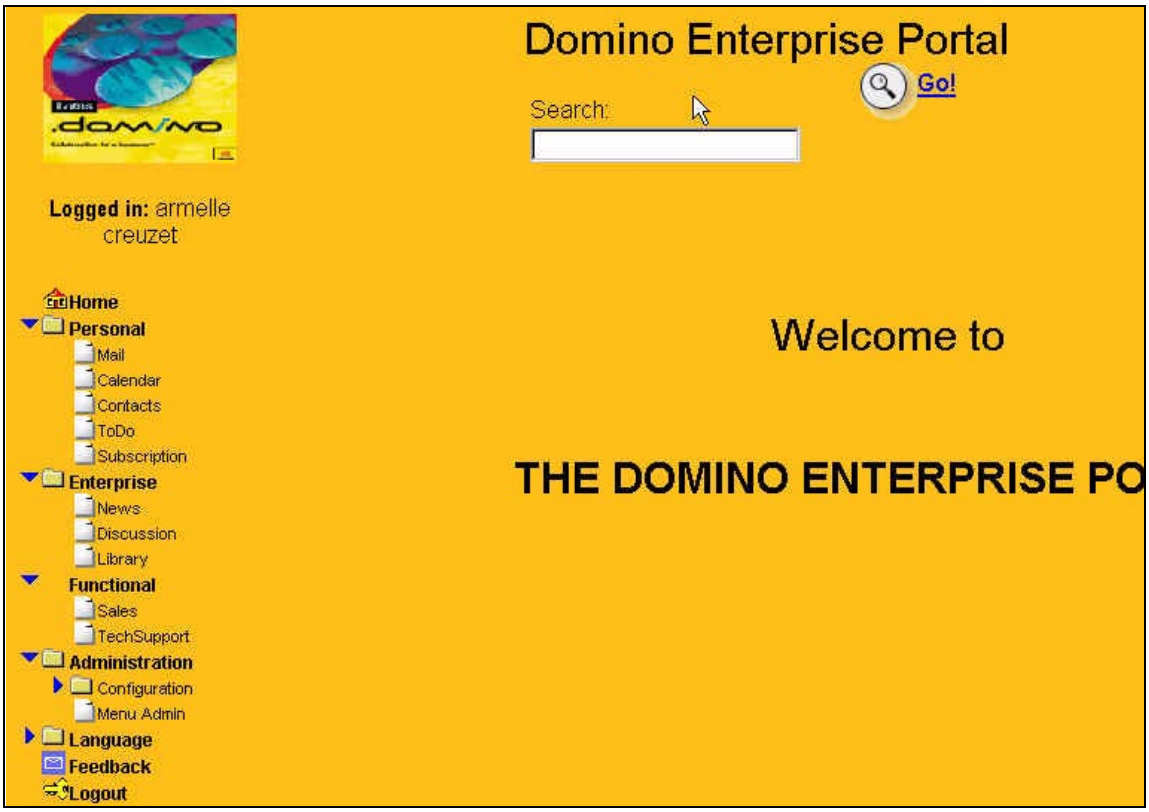


Figure 27. The portal homepage panel in English

According to Figure 27,the words to be translated are:

```
Welcome to
THE DOMINO ENTERPRISE PORTAL
Logged in
Home
Personal
Mail
Calendar
Contacts
ToDo
Subscription
Enterprise
News
Discussion
Library
Functional
Sales
TechSupport
Administration
Configuration
Menu Admin
Language
Feedback
Logout
```

Note

The Search frameset object, issued from another database (catalog.nsf), will not be translated. It needs the catalog.nsf translation.

4.2.5.1 Handling untranslated terms

In a Notes client, open the Portal S/390 glossary database, and select the **Untranslated terms** view. See Figure 28.

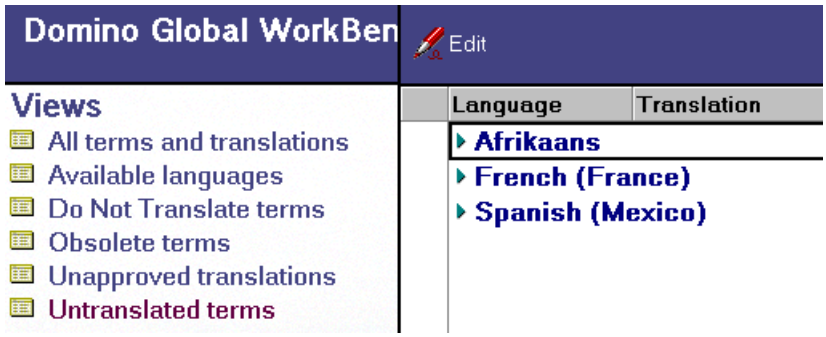


Figure 28. Untranslated terms view

The view is sorted by language. Collapse all the sections to ensure a translation to the correct language. Then expand the section you want to translate. Search for the items previously identified to be translated; see Figure 29 on page 70.

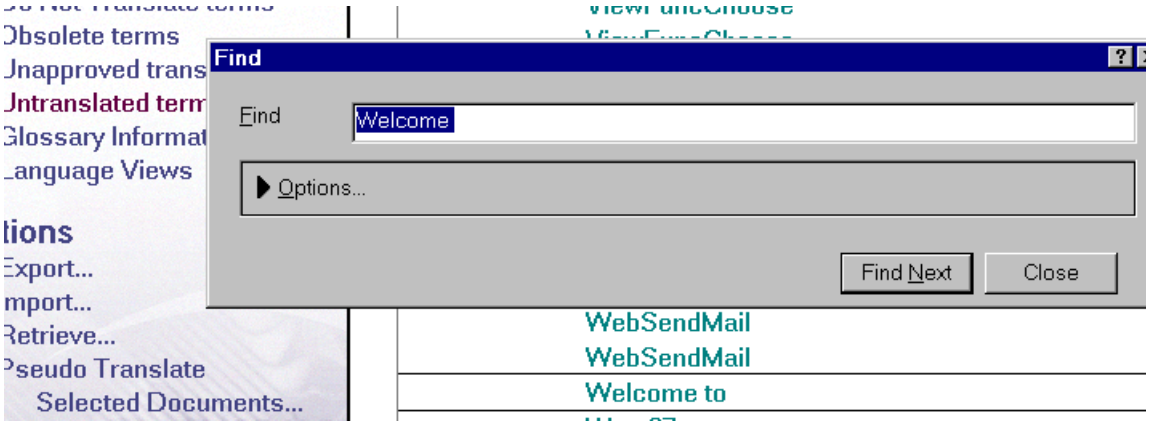


Figure 29. Searching for words to be translated

Edit the item. Double-click the **Welcome to** document. Translate the word(s) and click **Mark Translated**. See Figure 30.

Figure 30. Marking a document translated

On the next panel, mark this document **Approved**, then **Save** and **Close** the document. See Figure 31.

Figure 31. Marking a document approved

Notes

Be sure to translate the correct word and not another element of the design, especially if there are several instances of the word.

The translated words can be reviewed by opening the Glossary Language View, as shown in Figure 32.

Figure 32. Glossary language view

- Refresh the language database.

  Launch the Domino Global Workbench application and open the Portal S/390 project. From the dropdown databases names list located in the Tagged Database area, highlight the language database you want to refresh with the previously translated words. Click **Re-Build Language Database**; see Figure 33.

Figure 33. Re-build language database

Notes

There are three cases for which the language database will be rebuilt rather than updated.

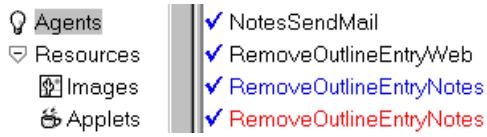
- Changing tagging or building selection
- Changing tagging or building options
- Making changes to the translations in the glossary

4.2.6 Managing design changes

Two methods are available to update all the languages databases by modifying only the source database.

1. Build synchronized databases. In this case, a change to one language database is automatically shown in all the other synchronized databases.
2. If design changes are made to the source database, the WorkBench causes the update of the tagged database and all associated language databases to reflect these modifications. The process is a two-stage one.
   a. Workbench identifies, by looking at the design elements date stamps, those that have been added/modified since the last translation. The application uses a color coding method to show the differences. The elements shown in black are the same in both source and tagged databases. The elements shown in blue have been added since the previous release. Those shown in green exist in both source and tagged databases, but they have been modified. Elements in red are present in the tagged database, but don't remain in the source.

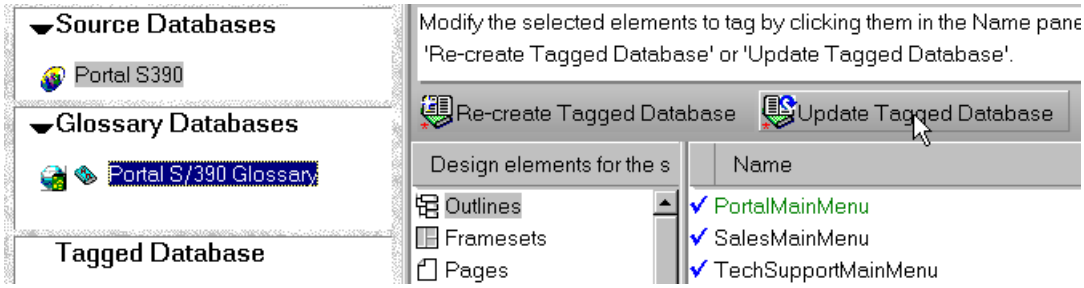
.



Elements that have been re-created, by copy/paste for example, appear in both red and blue color.

Identify the design changes to be implemented in the tagged database by selecting the colored elements listed by design name.

b. Update the Tagged Database by highlighting the source database and pressing **Update Tagged Database**.



Highlight the language databases and press the Update Language Database button. See Figure 34.

Figure 34. Update language database

The database now reflects the design changes.

4.2.7 Implementing the multilingual databases

Two outline entries per language database were added in the PortalMainMenu Outline design element; one entry for Notes clients, the second one for the Web clients. An icon picture was also added to the design. See Figure 35.

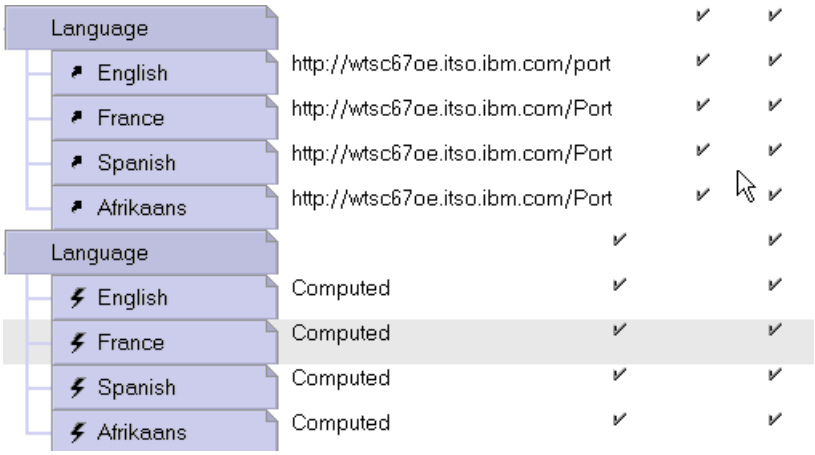


Figure 35. Outline entries for language database

1. Notes client entries are computed as type “action,” as shown in Figure 36.

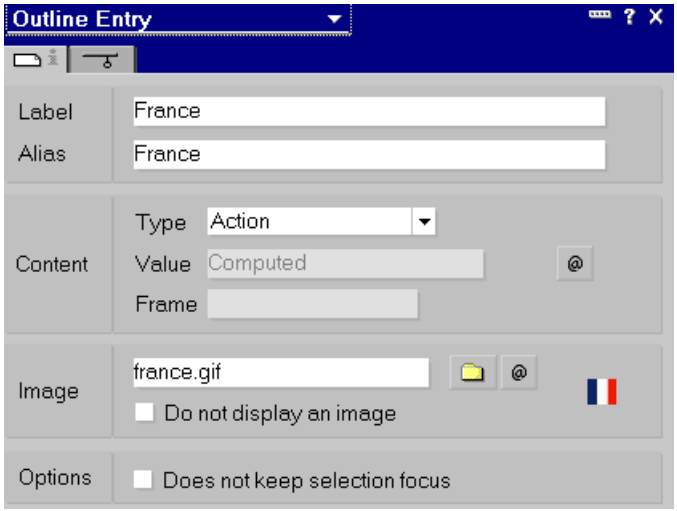


Figure 36. Outline entry with type “action”

The action formula is:

```
@Command([FileOpenDatabase]; "wtsc67oe" : "Portal_fr.nsf" )
```

Check the **Hide outline entry from Web Browsers** checkbox, as in Figure 37.

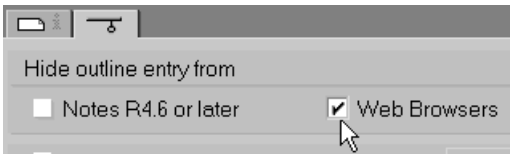


Figure 37. Hide from Web browsers

2. Web clients entries are URL links; see Figure 38 on page 75.

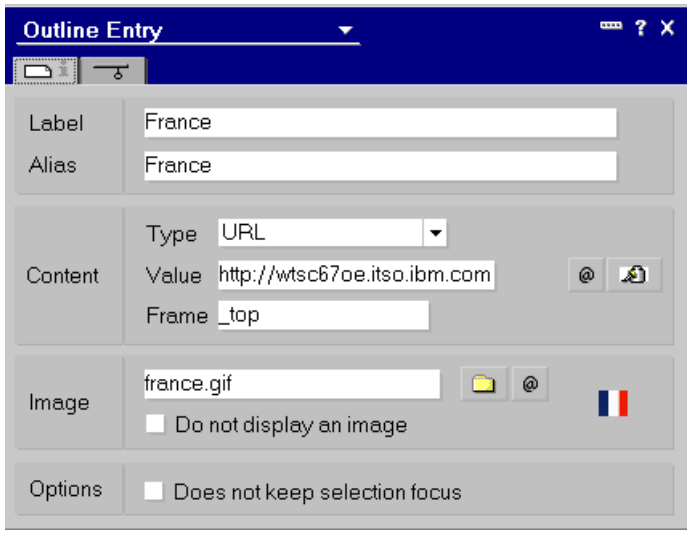


Figure 38. Outline entry

Check **Hide outline entry from Notes R4.6 or later**, as in Figure 39.

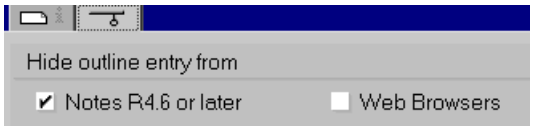


Figure 39. Hide from Notes R4.6

You can see the results of the multilanguage portal in Figures 40 and 41.

Figure 40. Portal in French

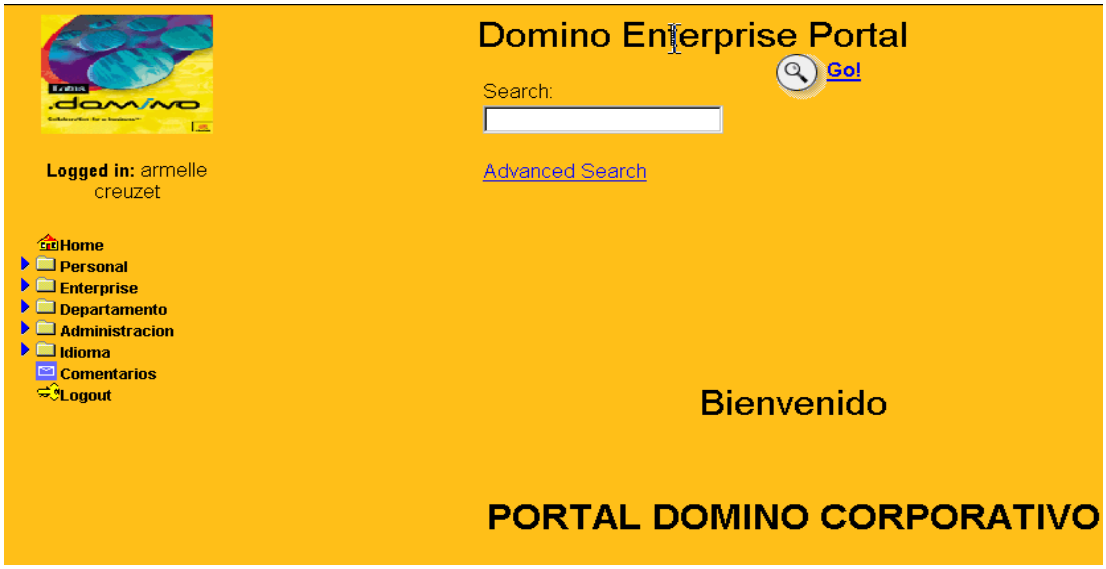


Figure 41. Portal in Mexican Spanish

Chapter 5. Portal security

This chapter describes how to achieve portal security via user authentication and well-planned access control.

5.1 Authentication and access control

Authentication is not the same thing as access control. Authentication deals with assuring the identity of the user, while access control deals with control of what a user can do. You cannot successfully deal with access controls without first dealing with authentication.

Should authentication not be an issue (for example, users can login anonymously to a News database), the ACL on the database must include the “Anonymous” entry to be set at least at the Reader level of permission.

5.2 Domino server and HTTP authentication

In this section we discuss the authentication process of the Domino server and its integrated HTTP server.

5.2.1 Domino server authentication

Whenever a Notes client or Domino server attempts to communicate with a Domino server for replication, mail routing, or database access, two security procedures use information on the client's ID to verify that the client is legitimate. Validation, the first procedure, establishes trust of the client's public key. If validation occurs successfully, authentication, the second procedure, begins. Authentication verifies the identity of the user. Authentication uses the public and private keys of the client and the server in a challenge/response interaction.

When a Notes user logs in to the Domino server, he is prompted for his password, which is stored in the ID file. After entering the password, his ID file is validated by the server. Then his notes ID file’s keys and certificates are compared to the ones on the server to assure the server of his identity.

For security reasons, Notes IDs should never be stored on a server or shared drive. To prevent and minimize unauthorized access to ID files, administrators should enable both the “Check Password” and Password Expiration fields in the person document (Figure 42 on page 78).

© Copyright IBM Corp. 2001

Password expiration will force users to change their Notes ID passwords every x days (a good period is normally 30 days). If more than one copy of an ID exists, the Check Password field will ensure that only the ID with the password corresponding to the entry in the Password Digest will be authenticated.

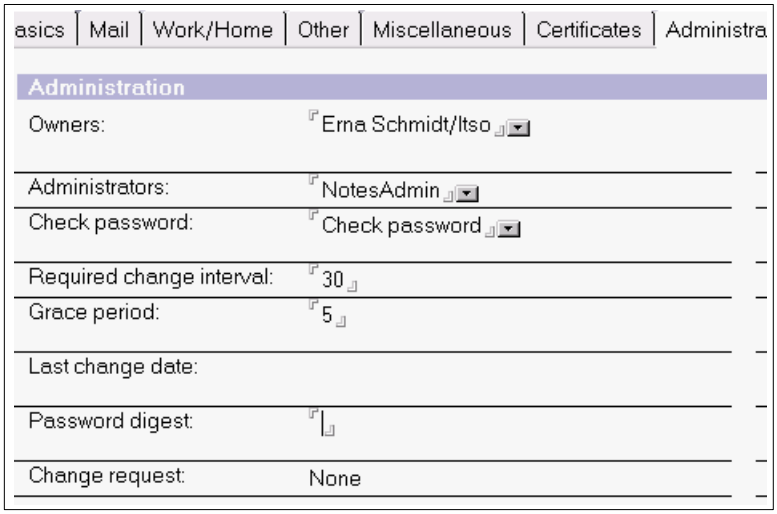


Figure 42. Notes ID password security

5.2.2 HTTP server authentication

Domino has an integrated HTTP server for browser clients, which contains a number of security features.

5.2.2.1 Name and password

You can select the level of restriction Domino uses when authenticating Web users in Domino Directories and LDAP directories. This makes servers less vulnerable to security attacks by refining how Domino searches for names and authenticates Web clients. Domino also uses this setting when a Java applet hosted on a Domino server authenticates users with the Domino IIOP protocol.

From the Domino Administrator, click the **Configuration** tab, and open the Server document. Click the **Security** tab. In the Web Server Access section, choose one of the following in the Web server authentication field:

- More name variations with lower security. Domino tries to authenticate users based on the name and password entered. This authentication method can be vulnerable to hackers who attempt to access a server

through a legitimate user account by guessing names and passwords. This choice is the default and is the Web server lookup behavior used in earlier releases of Domino. This option allows users to enter any of the following in the name and password dialog box in a Web browser: lastname, firstname, shortname, fullname, aliasname.

- Fewer name variations with higher security. This lookup technique is less vulnerable to attacks because a single authentication attempt does not produce as many matches, lessening the likelihood that a guessed password matches. This choice is recommended for tighter security. This option requires users to enter only the following in the name and password dialog box in a Web browser: full hierarchical name, common name, alias names (names other than fullname listed in the User Name field of the person document).

We recommend the last option: “Fewer name variations with higher security” to be used in all cases, be it for intranet or extranet authentication.

5.2.2.2 Authentication for Internet/intranet clients

Name-and-password authentication, also known as basic password authentication, uses a basic challenge/response protocol to ask users for their names and passwords and then verifies the accuracy of the passwords by checking them against those stored in Person documents in the Domino Directory. When set up for this, Domino asks for a name and password only when an Internet/intranet client tries to access a database on the server. Internet/intranet access differs from Notes client and Domino server access in that a Domino server asks a Notes client or Domino server for a name and password when the client or server initially attempts to access the server.

If you want to assign database access to an Internet/intranet client based upon Domino ACL security, you must create a Person document for that client in the Domino Directory. Clients who do not have Person documents are considered Anonymous and can only access servers and databases that allow Anonymous access.

Name-and-password authentication allows Domino to locate the Person document (if one exists) for the client accessing the server. Domino uses the Person document to identify the client. After the client is identified, access to databases can be determined.

You can use name-and-password authentication with either TCP/IP or SSL on any servers that run an Internet protocol—namely, NNTP, LDAP, POP3, HTTP, SMTP, IIOP, or IMAP. For each Internet protocol enabled on the server, you can specify the method of security. For example, you might enable SSL

for HTTP connections but require name-and-password security for NNTP connections that use TCP/IP. Or you might use name-and-password security with anonymous and SSL client authentication—for example, to allow users with SSL client certificates to authenticate using SSL client authentication and to allow other users to enter a name and password if they do not have an SSL client certificate.

If you are setting up name-and-password authentication for an HTTP server, you have an additional method for name-and-password authentication: session-based authentication. Session-based name-and-password authentication offers greater control over user interaction than basic name-and-password authentication and lets you customize the form in which users enter their name and password information. It also allows users to log out of the session without closing the browser.

The Domino Web Server Application Programming Interface (DSAPI) is a C API that lets you write your own extensions to the Domino Web Server. These extensions, or “filters,” let you customize the authentication of Web users. For more information about DSAPI and filters, see the Lotus C API Toolkit for Domino and Notes Release 5.0.3.

5.2.2.3 Authentication mechanisms

To set up name-and-password authentication for Web clients who have access to a Domino Web server, you can use one of the following two methods:

1. Basic name-and-password authentication, or
2. Session-based name-and-password authentication.

Basic name and password authentication

To set up basic name-and-password authentication for Web and Internet/intranet clients, you create a Person document for each user in the Domino Directory on the Domino server and assign an Internet password to each user. In the Server document, you must specify which Internet protocols require a name and password. And to allow these users access to databases, you must include them in database ACLs. Should authentication for certain databases on the server not be an issue, you should also specify that Anonymous access is allowed in the Server document.

Although Web and Internet/intranet clients require Person documents, they do not need Notes licenses since they do not use the Notes workstation software to access the server.

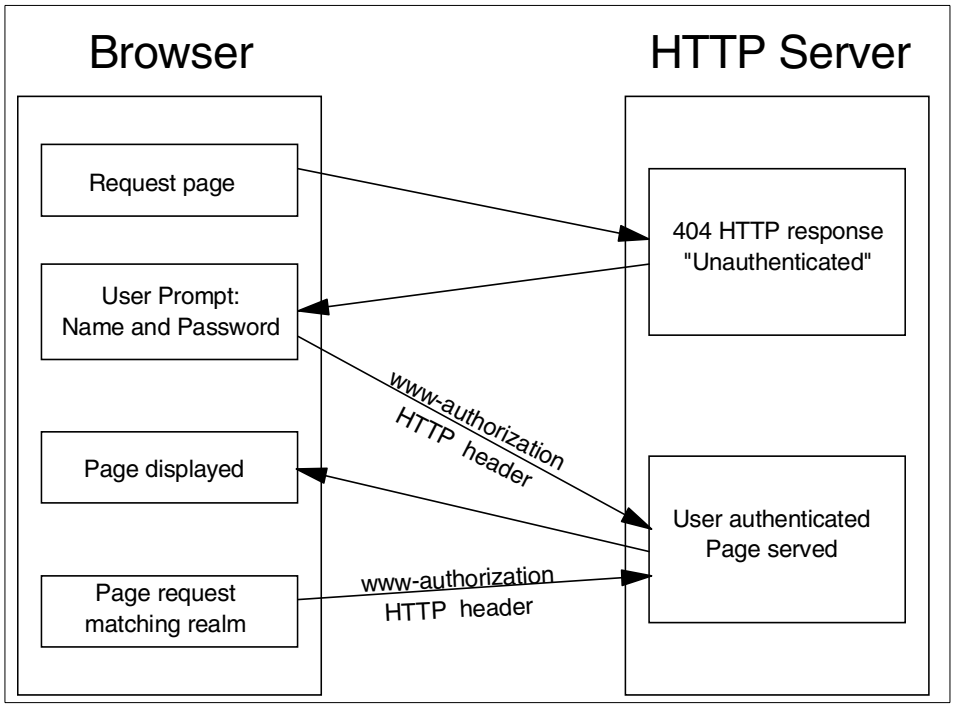


Figure 43. How basic HTTP authentication works

Figure 43 illustrates how basic HTTP authentication works. The client requests a page and when the HTTP server determines that the client did not send a valid www authorization HTTP header that matches the realm, it replies with a 404 HTTP response. This causes the browser to prompt the user for a username and password. The browser then sends the username and password as part of the HTTP header to the server. If the username and password is valid, the server will display the requested page.

The browser will continue to send the username and password in subsequent requests to the server if the same realm is accessed. It is important to note that the username and password is sent in the clear, and it is strongly suggested that server-side SSL is implemented when using this method of authentication in an internet environment. With server-side SSL, the connection will be encrypted and the username and password will be encapsulated within the encryption.

Session-based authentication

A session is the time during which a Web client is actively logged onto a server. Session-based name-and-password security includes additional

functionality that is not available with basic name-and-password security. You use the Server document in the Domino Directory to specify settings that enable and control session authentication.

Session-based authentication creates a temporary cookie that contains the username, password and random information in a hashed format. As the user traverses the site, responses for name and password are provided by the cookie. This cookie passes the user credentials for every database within the Domino site, thus alleviating concerns of realm-based authentication where the name and password are asked for every database residing in a different directory on the Domino Server. If you wish to retain realm-specific logins for different files or directories, session-based authentication should not be used. For more information regarding realm-based authentication refer to 5.2.2.4, “Realms” on page 89. Once a user logs into the Web site, the credentials are passed to every database hosted by the server. The user login information, however, is not shared across virtual hosts or virtual servers; it is based on the host name of the URL request.

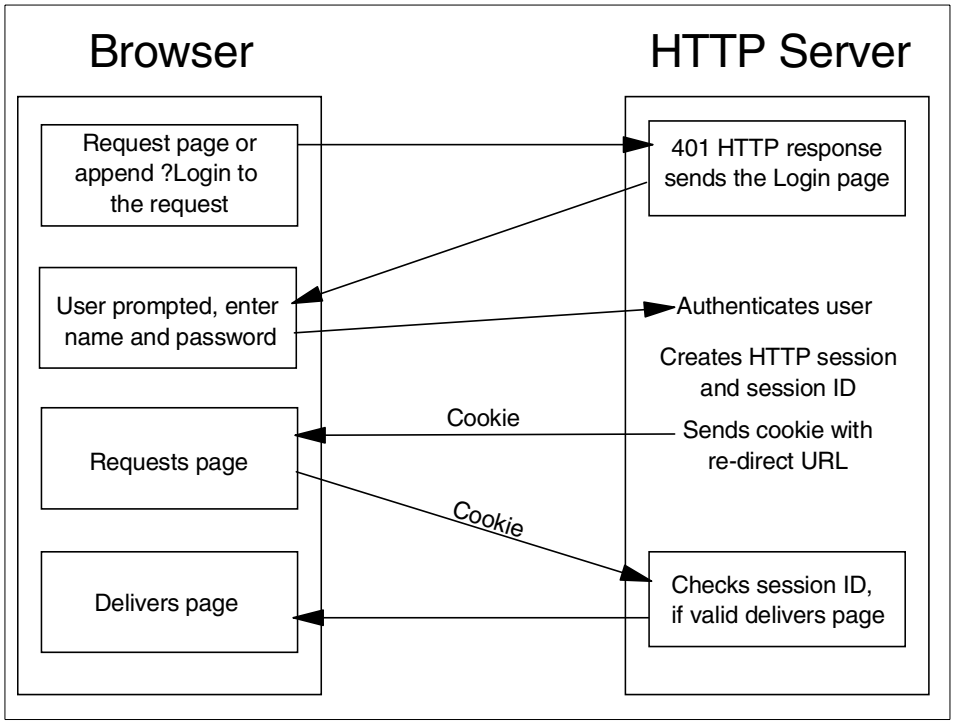


Figure 44. Establishing an HTTP session with session-based authentication

Figure 44 on page 82 shows how session based authentication works. The client requests a page and the server identifies that no cookie was sent. The server then sends a login page to the client. After the user supplies a username and password, the server authenticates the user. If the authentication is successful, the server creates a HTTP session and session ID. It then sends the cookie with a redirect URL to the client. The browser at the client will the issue a request for the redirect page to the server, together with the cookie ID. All subsequent requests form the browser will contain the cookie, and while the cookie is valid the server will serve the requested pages.

Again, it is important to note that the username and password is sent in the clear, although only once. It is strongly suggested that server-side SSL is implemented when using this method of authentication in an Internet environment. With server-side SSL the connection will be encrypted and the username and password will be encapsulated within the encryption.

To enable session-based authentication:

1. Open the server document.
2. Go to the **Internet Protocols**, **Domino Web Engine** tab
3. The first field: Session Authentication, can be changed between disabled (Realm based), Single server, and Multi Server. For the purpose of this portal we used Single Server authentication. For more information on Multi Server and Single Signon across multiple servers in a domain, refer to the Notes Administration Manual. Multi Server authentication is only available on Domino R5.05 servers and above.
4. On the server console issue the following command for your changes to take effect: `Tell HTTP restart`.

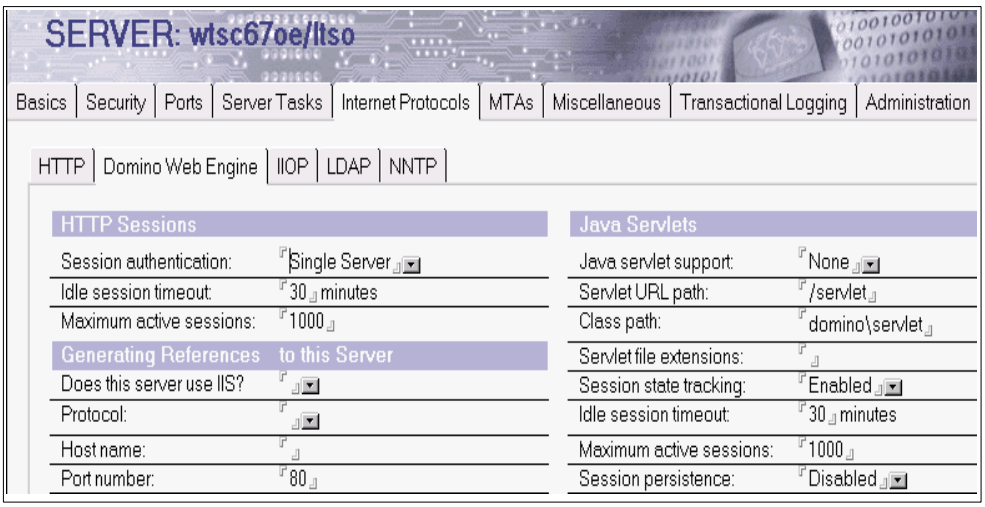


Figure 45. Enabling session-based authentication in the server document

Note

To use session-based authentication, Web clients must use a browser that supports cookies and the users' browsers MUST have cookies enabled.

Session-based authentication has several other features. You can customize the HTML login form your users see when prompted for authentication. Domino's customized login form is contained in the Domino Configuration database (DOMCFG.NSF). If this database does not exist, create it by doing the following:

1. Select **File -> Database -> New**.
2. Select your server name.
3. Type in the Database Name. Keep the filename as domcfg.nsf.
4. Select the template server and click **Show advanced templates**.
5. Select the Domino Web server configuration template (domcfg5.ntf).
6. Click **OK**.

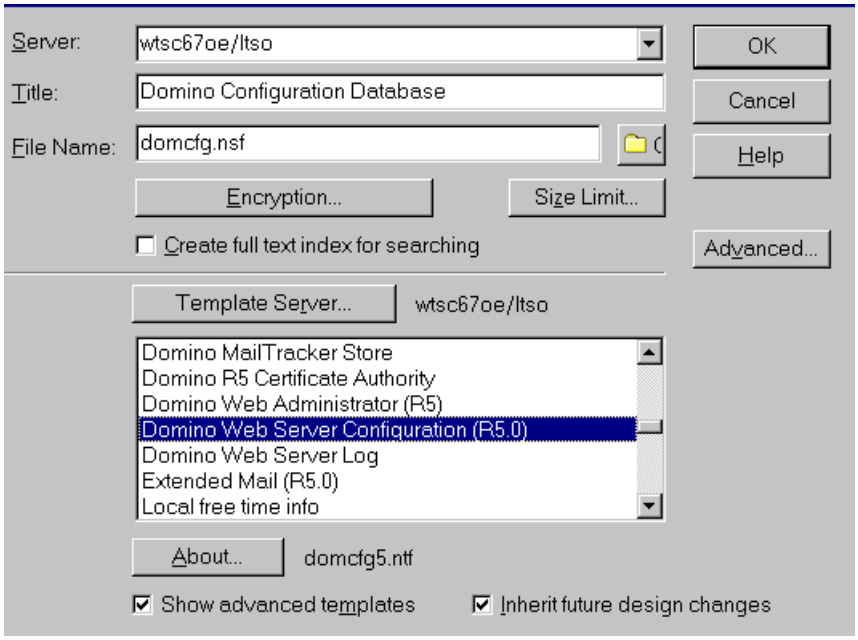


Figure 46. Creating the Domino configuration document

Once the Domino Configuration Database has been created you can change the login form for the Portal using the following steps:

1. Open the Domino Configuration database (domcfg.nsf).
2. Choose **Create, Mapping a Login Form**.
3. Enter the fields as depicted in Figure 47 on page 86.
4. Save the document.
5. Open the Domino configuration database in Domino Designer.
6. Open the $$LoginUserForm.
7. This form can be changed according to company standards (for example, background and text color, or an additional field such as login date).
8. Save the modified form and close the database.
9. The next time you login to the Portal the Login screen will show the modified login screen you created.

Note

It is always advisable to change the template file (domcfg.ntf) rather than the database (domcfg.nsf) itself. If you change the design of the nsf file, you run the risk that the design might revert to the old design the next time the design task runs on the server.

Another option would be to make a copy of the default login form. Remember to change the name of the login form in the mapping document to reflect the name of the new login form.

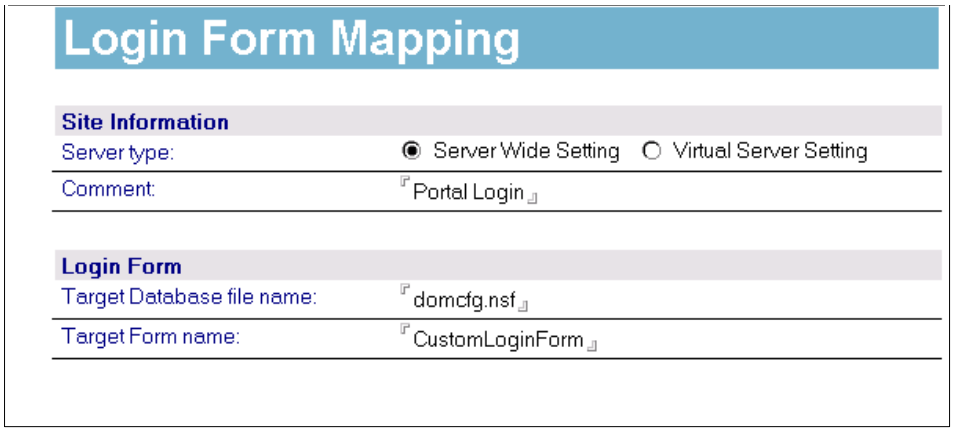


Figure 47. Creating a Login Form Mapping in the Domino Configuration Document

Session-based authentication also allows you to have user credentials expire. If you set a log out time period (found in the Server document below the Session Authentication field), the server drops the user's credentials after the specified amount of inactive time. In addition, you can now specify the maximum number of user sessions allowed on the server at the same time.

In addition to setting the log out time period in the server document, we recommend that you put a LOGOUT button on the portal for users handling sensitive information. In general, it is always a good idea to provide this functionality. If no LOGOUT button exists, users can log out of the session by appending “?logout” to the URL. If session-based authentication is not enabled, then the user must shut down the browser to log out of the Web server.

Note

If your servers are set up for round-robin DNS, do not use session-based name-and-password authentication. The servers cannot store the session information in memory when using round-robin DNS. In addition, if a server is restarted or crashes, session information is lost. Then the user must re-enter the name and password.

For our portal we chose session-based name-and-password authentication and created our own custom login form (see Figure 48) for the following reasons:

- Security
  - Username and password are only sent 'in the clear' once.
  - Users can log out.
  - Sessions can be timed out.
- Ease of use
  - User is only prompted once for credentials.
  - The active user sessions can be seen in the server console by using the `Tell http show sessions` command.
  - The login forms can be customized to your organization's needs.

Figure 48. Portal login form

We used the following javascript in the form onLoad event:

```
if (top.window.frames.length > 0)
top.location.reload(true);
else
{
document.forms[0].username.focus();
document.forms[0].username.select();
}
```

Because we used frames in our portal, the first javascript statement determines if we are in a frame; if so, we reload the login screen at the top level. This was done in order for the login screen to display in a full window when the session times out, and not in one of the frames.

The second part of the javascript gives the username field the focus so that the cursor is placed in the username field.

We also added the following javascript code in the onSubmit event of the form to ensure that the user provides a username and password in order to eliminate unnecessary network traffic:

```
var returnValue;
returnValue = true;
if (document.forms(0).username.value == "")
{
alert("You must supply a username.");
document.forms[0].username.focus();
document.forms[0].username.select();
returnValue = false;
}
if (returnValue == true && document.forms(0).password.value == "")
{
alert("You must supply a password.");
document.forms[0].password.focus();
document.forms[0].password.select();
returnValue = false;
}
return returnValue;
```

The reset button was added with the following Pass-Thru HTML:

```
<input type='reset' value='Reset'>
```

The purpose of the reset button is to clear all fields on the login form.

To enable a user to log out of our portal, we added an entry in our menu which sends the user back to the login screen. This can be achieved by

appending `?Logout` to the URL request. If you want to show another screen for the users, append `?Logout&redirectTo=/homepage.nsf` to the URL request, where homepage.nsf is the name of the database to open.

5.2.2.4 Realms

You can configure Domino Web sites so that Domino authenticates and asks Web users for their credentials only once when they access different locations. Like other Web servers, Domino adheres to the HTTP authentication model. When a user accesses a page on a Domino Web site, the browser keeps track of user credentials, based on the realm that the Domino server sends to the browser. A realm is a string, which is typically a URL path, that the server sends to indicate the location, or path, for which the user has been authenticated.

For example, if your server name is `www.acme.com`, then `www.acme.com` is the top-level realm and `www.acme.com/doc`, `www.acme.com/hr`, and `www.acme.com/marketing` are the lower-level realms. If a user authenticates with the server when accessing the home page for `www.acme.com`, then the user is authenticated for `www.acme.com` and all lower-level realms.

However, if the user accesses `www.acme.com/doc` first, enters a name and password and is authenticated, and then accesses `www.acme.com/hr`, Domino prompts the user for credentials again. This second prompt occurs because the browser examines the list of realms for which Domino has successfully authenticated the user and finds `www.acme.com/doc` in the browser realm list. Since `www.acme.com/hr` is not a subdirectory of `www.acme.com/doc`, Domino requires the user to enter credentials again.

To prevent users from being prompted multiple times for their names and passwords, direct them to access and authenticate with the highest level realm that they need to access. This way, Domino asks users for their credentials only once during the browser session.

If a Web site includes a link to a site on another server and that site requires authentication, users will be prompted again for their credentials.

Note

It is a good idea to create at least one high level realm, regardless of whether basic or session-based authentication is used. We used framesets in the creation of our portal. We found that, even with session-based authentication enabled, frames within framesets that referenced databases with no access for default and anonymous readers displayed the login page in each one of the frames. The only way we could get past that was to create a realm document for our server, as shown in Figure 49 on page 91.

To create a realm for your server:

1. Go to the Domino directory (names.nsf) and open the server document in edit mode.
2. Click **Web, Create Realm**.
3. The server name is already filled in. Use the IP address field only if you are using Virtual servers
4. For Path, enter the full hostname of the server. This is the highest level realm and will ensure that the user only has to enter their credentials once. Ensure that this host name has been correctly defined in the DNS.
5. Realm returned to browser when access is denied should contain the text that describes the location to the user whenever there is an authentication failure at the location. The text is displayed in the browser's authentication dialog.
6. To enable the realm, enter the command `tell http restart` on the server console.
7. Web Realm documents can be displayed by going to the Domino Administration client, Configuration tab. The document is displayed under **Web, Web Server Configurations** as a response to the server document.

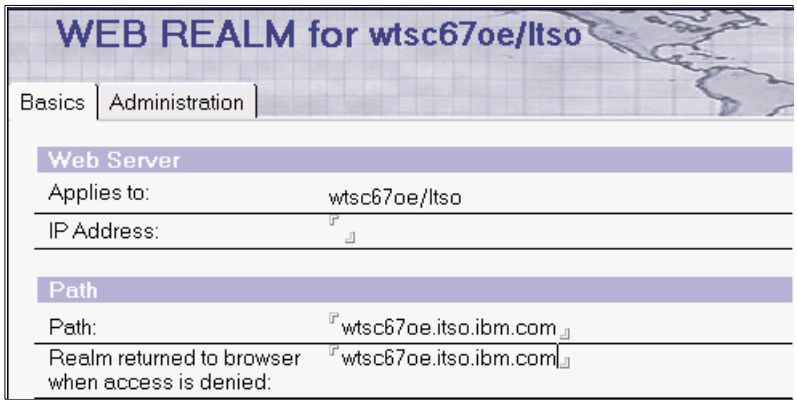


Figure 49. Web realm document

5.2.3 Secure Sockets Layer

Secure Sockets Layer (SSL) is a security protocol that provides communications privacy and authentication for Domino server tasks that operate over TCP/IP.

The TCP/IP protocol forms the underlying foundation for scalable Internet communications. Unfortunately, TCP/IP is, by its very nature, unsecured. However, using the SSL protocol in conjunction with TCP/IP brings reasonable first-level security to this previously unsecured environment.

Perhaps the most important function that SSL brings to standard TCP/IP and HTTP sessions is the ability to mutually authenticate servers and clients. By enabling SSL, you can establish trusted relationships between clients and servers using certificates and digital signatures. While this ability has long been available to Notes users, it was a ground breaking development in security across TCP/IP.

The Domino server is capable of providing a very similar type of authentication for browser clients as it does for Notes clients using SSL and X.509 certificates. These certificates are stored in the user's Web browser as well as on the server. While these provide a secure way of authenticating with the server, they tend to be a high maintenance solution and may be expensive to implement if you intend to use third-party certification to protect both the user and the server from each other. (Several companies offer this service, including Verisign and GlobalSign).

For many sites, the level of security offered by X.509 certificates and SSL is not necessary. These sites can use “basic authentication,” which simply

requires a name and password. All the Domino server needs for basic authentication is a Person document for each user in the Domino Directory with the User Name, Last Name, and Internet Password fields completed.

5.2.3.1 How SSL works

SSL works by providing three security services, each of which use public-key encryption techniques.

The first and most basic function of SSL is message privacy. SSL offers feature support so applications can exchange and authenticate user names and passwords without exposing them to eavesdroppers. Hackers can use IP sniffers to download copies of all packets that pass between a client and server during a session. This information is then available in an unencrypted, cleartext format. With SSL, however, all transmissions following the initial handshake are encrypted to prevent such transmissions from being read.

Second, the SSL protocol ensures that messages between the sender and receiver have not been tampered with in any way between the two points. This maintains session integrity and keeps a secure channel open between the client and server.

The final and most important function of SSL is mutual authentication. This is the process by which the client and server can convince its peer of its identity through the exchange of X.509 certificates. A server's identity is coded in the form of a public-key certificate that is exchanged during the SSL handshake.

5.2.3.2 The SSL handshake

SSL was designed to make its security services as transparent as possible for the end user. Typically, a user follows a standard HTML page link that connects automatically to an SSL-capable server.

Most SSL-capable Web servers take the SSL connection on a different port (443 by default) from standard HTTP requests (port 80). When the application establishes a port connection, it initiates a handshake that establishes the SSL session. This URL is then prefixed with http**s** instead of http.

From this point forward, all communications between the client and server are encrypted and message integrity checks are performed until the SSL session has been disconnected. In an SSL session, the handshake only occurs once.

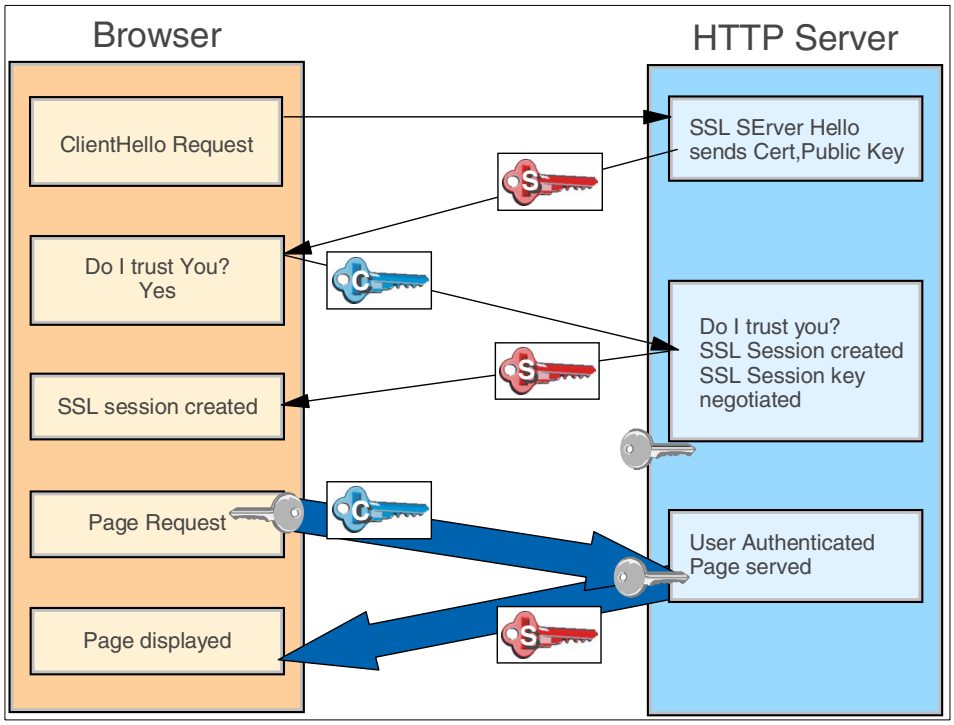


Figure 50. SSL handshake

Three events occur in the execution of the SSL handshake.

- First, client and server (or server and server) exchange X.509 certificates to prove their identity. The certificate is checked against expiration dates as well as for evidence of the non-tampered signature of a trusted third-party authority.
- During the second handshake event, the client randomly generates a set of keys to be used for encrypting the messages. This, in conjunction with the server's public keys, creates a total of four keys, two of which are used to encrypt the data transmission. Basically, the client encrypts a cipher key with the public key that it received from the server. The client then sends the encrypted message to the server. The server now decrypts that message with its private key. If all is well, the keys that were exchanged can now be used throughout the session to protect the data from lurkers.
- As a part of the mutual negotiation phase, the client and server determine the hash function (for integrity) and the message encryption algorithm (for encryption). In this process the server asks the browser to provide a list of

all possible ciphers. The server then encrypts the information using the strongest allowable cipher that it holds in common with the client.

5.2.3.3 SSL security benefits

1. Data is encrypted to and from clients, so privacy is ensured during transactions.
2. An encoded message digest accompanies the data and detects any message tampering.
3. The server certificate accompanies data to assure the client that the server identity is authentic.
4. The client certificate accompanies data to assure the server that the client identity is authentic. Client authentication is optional and may not be a requirement for your organization.

Domino supports SSL versions 2.0 and 3.0 when using HTTP, POP3, IMAP, LDAP, and NNTP. It can be set up on a protocol by protocol basis. X.509 is the open standard certificate format for inter-application authentication. You can now use either traditional Notes certificates and/or X.509 certificates (which then allow interoperability with other certificate authorities and standards, such as S/MIME).

5.2.3.4 The role of certificate authorities

The SSL handshake manages the exchange of certificates and digital signatures. Domino fully supports industry-standard RSA encryption and CAs (certificate authorities).

The role of certificate authorities is to enhance mutual trust. In street language, they stand as a third party between two other parties to verify that one of the parties is not pulling the wool over the other's eyes. The process works like this:

1. The certificate authority signs a certificate (yours, we will say) with their private key to validate it.
2. You send me your certificate.
3. I use the public key given to me by the certificate authority to check that your certificate was indeed issued by them and that it has not been altered since it was issued. Of course, to do this, I have to be able to find the public key for your certificate authority in a reservoir of keys that are managed by my server.

Notes certifiers are a true certificate authority, functioning within a Domino environment. Consequently, understanding the role of a Notes certifier will help you to understand and interact with Internet-centric certifiers.

5.2.3.5 Domino, SSL and Notes security

With Domino, you can enable SSL both on the Domino server and client-browser side (such as Web Navigator).

You complete the setup process regardless of whether you request a server certificate from a Domino or third-party CA. For the purpose of this redbook, we set up Domino as the certificate authority.

The steps involved to set up a secure environment with both server and client (browser) SSL certificates using a Domino certificate authority are:

1. Create the Domino certificate authority database and the keyring for the CA. This keyring holds the public and private key of the CA. The private key is exclusively used to sign server and client certificates. The public key of the CA is signed with the CA's private key. This signed public key is used to provide a certificate flagged as a "trusted root" to be installed in server and clients to establish a chain of trust.
2. Create the Domino server's keyring and have its public key certified by the Domino CA. This will require setting up a Domino Server Certificate Administration database to manage the server's keyring file.
3. Request a client certificate(s) for the browser(s) and user(s) to access the server. If you are using certificate-based authentication (not just SSL connections), store the client certificates in the Domino directory for authentication.
4. The steps outlined require access by both a browser and a Domino Administrator client.

For more information and detailed instructions for setting up SSL on Domino, refer to the redpaper REDP0046: *Domino Certification Authority and SSL Certificates* by Mike Bartlett.

Our SSL implementation does not in any way diminish the availability of, or our commitment to, Notes-specific security. Existing Notes network protocols retain full support. In fact, you can and should continue to use the Notes security, authentication, and encryption model wherever possible. There are two reasons for this:

1. First, in an intra-Domino environment, the Notes certification model is richer than SSL. It employs a deep, hierarchical naming structure which simplifies both the issuance of certificates and system verification of those

certificates. There is a chain of trust both up and down within a certificate tree and across trees to other chains that have been verified for trust. In effect, Notes maintains a semantic certification network, where the Internet model is merely syntactical.

2. Second, SSL in no way affects your management of application access, using ACL and other mechanisms. SSL is indifferent to who actually accesses an application, once the veracity of the connection itself is established. This is critical. After all, authentication and encryption are merely a minimum, not a maximum, condition for mission-critical computing. Within a Notes environment, we can open or close access at the field level.

Therefore, Internet security solutions can be viewed as a subset, not a superset of onboard Notes security. Over time, Internet security might develop enough to match or surpass today's Notes security arsenal.

5.2.3.6 Options: SSL authentication

We have two options in setting up SSL: server-side SSL, or server-side SSL plus Session authentication.

Both options will encrypt the connection between server and client, but if you have client authentication SSL-enabled, the HTTP session will authenticate and verify the user; whereas with server-side only SSL, the authentication is done with basic name and password authentication in the Domino directory.

Client authentication differs from server authentication with Domino basic password authentication in significant ways. Table 1 on page 97 shows a comparison between the two options.

Depending on your organization, you might benefit from some of the client authentication's seeming lack of functionality. For example, if your users continually forget or lose their passwords, the fact that they do not have to enter a name and password might be beneficial. In addition, by not allowing the client certificate to move to another computer, you might further secure data since a hacker must have explicit access to the computer to impersonate a user.

Table 1. SSL client versus server-only authentication

	SSL Client authentication	SSL Server Authentication with basic password authentication
Requires the user to enter a name and password	No	Yes
Encrypts transactions over the network, identifies the server, validates messages being sent	Yes	Yes
Validates user identity	Yes	Not validated by a third party; only as trusted as the physical security imposed on the name and password
Portable to other computers	No browser software contains the certificate	Yes; user can enter the name and password on any machine
Name and password encrypted over the network to prevent eavesdropping	n/a	Yes

When you might not need client authentication

Client authentication may not be the security solution for everyone. Before you commit to using client authentication at your organization, here are some things to think about:

- Do you need to gather information on users accessing your server or control access to databases based on the user identity? If not, don't bother with client authentication. If you set up server authentication, the data transmitted is still secured and the server identity is validated for the user.
- Both you and your users accessing the server must perform steps to set up client authentication. If you have a Web site that you want users to access easily and the previous item doesn't apply, then you probably do not want client authentication. Users might get discouraged and leave your site if you require them to get a client certificate before accessing it.
- If it is a requirement at your organization that clients must use multiple computers and you do not want to set up client authentication for the user on each computer, then use SSL server authentication with Domino basic password authentication instead. This lets the user identify themselves

without requiring that they use the same computer each time they access the server.

Thus, depending on your organization's specific need, server-side SSL with basic authentication should do the job in most cases.

5.3 Database access control

After the user has been authenticated, be it by the Domino or the HTTP server, his level of access to information on the server is determined by the access control set individually on databases, views, forms, and even fields.

For the purpose of our portal, we used only database-level access control.

5.3.1 Access control lists

Every database includes an access control list (ACL) which Domino uses to determine the level of access that users and servers have to a database. Acceptable entries in the ACL include user, server, and group names, database replica IDs, the Anonymous name for Internet user access and anonymous Notes user access, Internet clients' user and group names, and alternate names.

To allow general access to a database, you can enter hierarchical names with a wildcard character (*) in the ACL. You can use wildcards in the common name and organizational unit components. All users and servers whose hierarchical names include the components that contain a wildcard receive the specified access to the database (for example, */ITSO with Reader access) will allow all users certified with the ITSO certifier Reader access to that specific database.

You can add to an ACL the names of any individuals with certified Notes user IDs, or Internet users who authenticate using name-and-password or SSL client authentication.

If the Notes user name you are adding is in a different hierarchical organization than that of the server that stores the database, you must enter the full hierarchical name for the user; for example John Smith/Sales/Acme.

If the Notes user name of the person you are adding and the server that stores the database are in the same hierarchical organization, you need to enter only the common name component of the hierarchical name; for example, John Smith. To ensure tighter name security, use the full hierarchical name.

For Internet clients, you must enter the first name that appears in the User name field of the Person document. The -Default- access and Anonymous entry in the ACL must be set to No access; otherwise, Notes gives the Internet client the -Default- or Anonymous access instead of the access specified for the client.

Any user or server that accesses a remote server anonymously is known by the name “Anonymous” at that remote server. Anonymous access is given to Internet users and Notes users who have not authenticated with the server.

5.3.2 Roles

As an additional level of security, one can also create one or more roles, and assign each user to a role. In the portal database, we used roles as a mechanism to provide a basic level of personalization. Refer to Chapter 4, “User personalization” on page 55 for more details.

5.3.3 Groups

A group is a list of users or servers that require the same access to a database. You can add a group name—for example, Sales—to the ACL to represent multiple users or servers that require the same access. Any group listed in the ACL must exist in a Domino Directory before it is added to a database ACL. Or it must reside in an LDAP Directory that has been configured for group expansion in the Directory Assistance database.

Using Groups in an ACL is a much better way to provide security than using individual names.To create a new group for inclusion in the ACL, you must have Manager access to the database.

Using a group in the ACL offers the following advantages:

- If you need to change the access level for several users or servers, you can do so once for the entire group. If a group is listed in more than one ACL, you can modify the list of members from a central location, the Domino Directory or the LDAP Directory—that is, rather than add and delete names in multiple databases.
- You can use groups to let certain users control access to the database without letting them change the design. For example, you can create groups in the Domino Directory for each level of database access needed, add the groups to the ACL, and allow specific users to own the groups. If these users don't have Manager or Designer access in the ACL, they can't modify the design of the database.

- Group names can reflect the responsibilities of group members or the organization of a department or company.

5.3.4 Access control—best practices

Server security is a major issue and should not be taken lightly. To control access to information on a server one should enable the highest level of security possible to the databases residing on your server.

1. *Always* change default access to “None”. Should you need Anonymous access to the database, include the “Anonymous” entry in the ACL.
2. “Anonymous” access should mostly be limited to Reader access. Author access is the highest level of access that should be allowed, and then only for databases such as newsgroups or discussions.
3. Whenever you have more than one user with the same access to a database, create a group in the Domino Directory rather than using individual names in the ACL. It is easier and much neater to manage.
4. Try to always include a NotesAdmin group containing the names of the Domino server administrators (except in cases of extremely sensitive data) with Manager access to all ACLs. This helps the administrators to quickly fix problems or restore corrupt data and allows them to change the ACL quickly in cases of a security breach.
5. Never leave the User Type of an ACL entry as “Unspecified.” Change it to the proper type for the group or user, for example Person, Server, Person Group, or Server Group.
6. The highest level of access any user should have to a database is “Editor” access. Only administrators of a database and the administration server should have “Manager” access. Servers other than the administration server which host replicas of the database should be included in a group with type “Server Group” and “Editor” access.
7. Never use “Designer” access on a production database. Designer access is only necessary on databases being developed or changed and changes should never be made on the fly on a production server. Rather use templates to make changes and test the changes before having production databases inherit the design from the template.
8. Set an Administration Server on the ACL Advanced tab—especially when you have multiple replicas of a database. This will ensure that the database ACL can only be changed on the server specified as the Administration server. Using this in conjunction with the setting “Enforce a consistent access control list” field on the Advanced tab will ensure that

databases will not be able to replicate if the ACL is changed on a server other than the Administration server.

9. The highest level of access any user, except for administrators, should have to your Domino Server Directory (names.nsf), is Reader access.

We realize that there are always exceptions to the rules. But these exceptions and their implications should be carefully considered before being allowed.

Part 3. Applications for the portal

© Copyright IBM Corp. 2001

Chapter 6. Integrating personal and enterprise pages

In the next two chapters, we explain the technical elements that are applied during the creation of portal applications.

We begin by describing the development of the individual elements that compose the different portal pages in both the Notes and Web client environment, and then we discuss the integration of these elements into their corresponding portal pages.

Once we have completed the construction of all the pages, we show how to integrate the pages into the kernel portal application that groups together all the elements.

6.1 Personal portal pages

As we discussed in Chapter 3, “Kernel portal architecture” on page 29, the personal portal consists of pages that contain a user’s personal mail, calendar, contacts, to do list, subscriptions, and Sametime chat option. Each page provides an application related to the current user.

We had to make a distinction between the Notes client environment and the browser environment for the personal pages, because the Web browser environment does not cater to the concepts of location documents or Notes user IDs. Therefore, we had to use an agent to collect the user’s information from the server’s Domino Directory and return the user’s mail file location.

We explain the content of each of the personal pages beginning with the Notes client environment. We then proceed to the Web browser environment.

Figure 51 on page 106 shows an overall view of the outline entries for the personal portal pages. These entries will be displayed in the menu of the kernel portal.

© Copyright IBM Corp. 2001

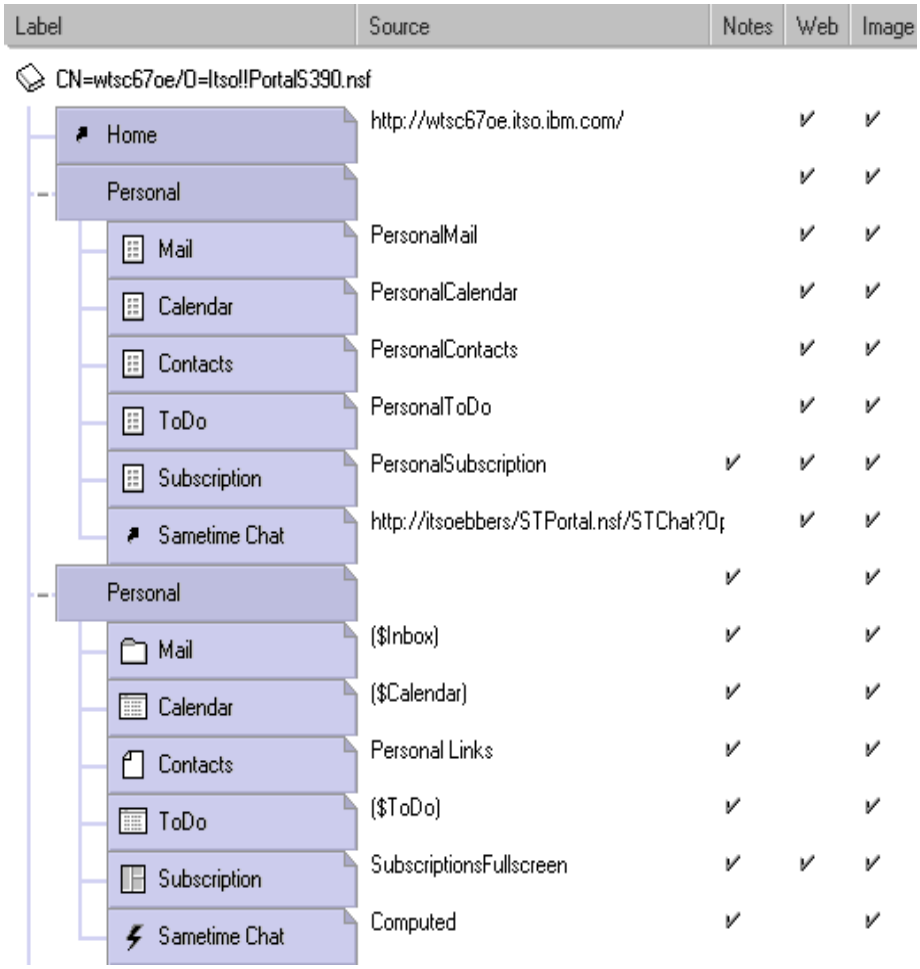


Figure 51. Outline entries for the personal portal

6.1.1 Notes client

We implemented this portal with both a Notes client and a Web browser end-user interface. First we discuss the Notes client.

6.1.1.1 Personal mail

This page contains an e-mail application. To include this application in the menu using a Notes client is relatively easy. We simply create a link to the user's personal mail database, which already exists in the Lotus Domino infrastructure.

The following steps describe the creation of the link:

- Open the Designer and at the design pane click the **Outlines** bookmark.
- Select the **PortalMainMenu** outline and double-click to open it.
- Add an outline entry and indent it underneath the Personal entry for the Notes client.
- Name this entry `Mail` with alias `Mail`.
- Choose **Named Element** from the drop-down list and choose **Folder** in the drop-down list field to the right.
- Click the folder icon and the Locate Object dialog box opens.
- Click the drop-down list arrow in the Database field and select **-Users mail file-**
- Click the drop-down list arrow in the Folder field and select **($Inbox**).
- Click **OK** to close.
- Set the frame to **_blank** to open the mail page in a separate window.
- We also included an image for the menu entry, but this is optional and only for presentation purposes.
- Remember to check the Web Browsers checkbox in the Hide outline entry from tab.
- Close the outline entry box and save the outline.

6.1.1.2 Personal calendar

The Notes client mail application also contains calendaring and scheduling features that can be used for personal time management, meetings and appointment scheduling, or to delegate work to others. The Calendar View provides a desktop calendar for viewing scheduled appointments.

In this link, we will allocate the personal calendar application corresponding to the currently logged-in portal user. Similar to the mail application, we create a new outline entry and call it `Calendar` with alias `Calendar`.

Follow these steps to create the link:

- Choose **Named Element** from the drop-down list in the outline entry box, and choose **View** in the drop-down list to the right.
- Click the folder icon and the Locate Object dialog box opens.
- Click the drop-down list arrow in the Kind of object field and select **View**.

- Click the drop-down list arrow in the Database field and select **-Users mail file-**.
- Click the drop-down list arrow in the View field and select **($Calendar)**.
- Click **OK** to close.
- Set the frame to **_blank** to open the calendar page in a separate window.
- Include an image if required.
- Remember to check the Web Browsers checkbox in the Hide outline entry from tab.
- Close the outline entry box and save the outline.

6.1.1.3 Personal contacts

For the personal contacts link, we created a page named Personal Links. In this page we included the links to the Enterprise Contacts, External Contacts and Customers. The layout of the Personal Links page is shown in Figure 52.

Figure 52. Personal links page layout

The following shows the internal sequence that takes place when a user clicks on one of links:

- “Ent. Contacts” -> HotSpot Resource Link, Content: Named Element -> View -> People (Database: -User’s address book-).
- “External Contacts” -> HotSpot Resource Link, Content: Named Element -> View -> Contacts (Database: -User’s address book-).
- “Customers“ -> HotSpot Resource Link, Content: Named Element -> View -> OrgnizationStructure (Database: Customers).

In all of these HotSpot Resource links, we set the frame to _blank to open the link in a new window. This give the user the choice of keeping the link open in a separate window or closing it, without disturbing his view of the kernel portal.

Follow these steps to make the corresponding link in the menu outline:

- Create a new outline entry and call it `Contacts` with alias `Contacts` and indent it underneath the Personal outline entry for the Notes client.

- Choose **Named Element** from the drop-down list in the outline entry box, and choose **Page** in the drop-down list to the right.
- Click the folder icon and the Locate Object dialog box opens.
- Click the drop-down list arrow in the Kind of object field and select **Page**
- Click on the drop-down list arrow in the Database field and select **-Current database-**.
- Click the drop-down list arrow in the Page field and select **PersonalLinks**.
- Click **OK** to close.
- We wanted the page to open in the EnterpriseBody frame, so we set the frame to EnterpriseBody.
- Include an image if required.
- Remember to check the Web Browsers checkbox in the Hide outline entry from tab.
- Close the outline entry box and save the outline.

6.1.1.4 Personal to do list

This link opens the personal to do list of the user that is logged into the portal.

Follow these steps to make the corresponding link:

- Create a new outline entry and call it `ToDo` with alias `ToDo` and indent it underneath the Personal outline entry for the Notes client.
- Choose **Named Element** from the drop-down list in the outline entry box, and choose **View** in the drop-down list to the right.
- Click the folder icon and the Locate Object dialog box opens.
- Click the drop-down list arrow in the Kind of object field and select **View**.
- Click the drop-down list arrow in the Database field and select **-Users mail file-**.
- Click the drop-down list arrow in the View field and select **($ToDo)**.
- Click **OK** to close.
- Set the frame to **_blank** to open the ToDo page in a separate window.
- Include an image if required.
- Remember to check the Web Browsers checkbox in the Hide outline entry from tab.
- Close the outline entry box and save the outline.

6.1.1.5 Personal subscriptions

In order to complete the subscriptions link, we must create and configure the subscription database. The Lotus Domino infrastructure contains a standard template named Subscriptions. We used this template to create the subscriptions database.

Subscriptions database configuration

Subscriptions work in conjunction with the headlines database. The headlines database enables users to stay informed of current events both within their company and on the Web. The headlines database lets users subscribe to databases that are of interest to them and then receive notification when a posting meets their subscription profile. When you design a database you can enable it to be subscribed to by users. The following list contains important considerations when enabling a database for subscriptions.

- The database must reside on a server that allows subscription monitoring. The Domino administrator must enable subscriptions on the server document. See *Administering the Domino System*, Lotus part numbers CT7VHNA and CT7VINA, for more information on enabling subscriptions at the server level.
- The database must enable headline monitoring. This is enabled on the Advanced tab of the Database Properties box. It is enabled by default. For more information about performance issues with headline monitoring, see “Prevent headline monitoring” in *Managing Domino Databases,* Lotus part number CT73MNA. Other topics from this manual are referenced below.
- The database must have a default view specified. For more information on specifying a default view, see the topic “Default views.”
- Optionally, the database can contain a subscription form. You can design one yourself or copy the subscription form from the headlines.nsf database installed with the Notes client. To copy the subscription form from the headlines database to your database, see “To copy an individual design element”. If you are creating a subscription form from scratch, you must use a formula field. For more information, see the topic “Formula fields.”

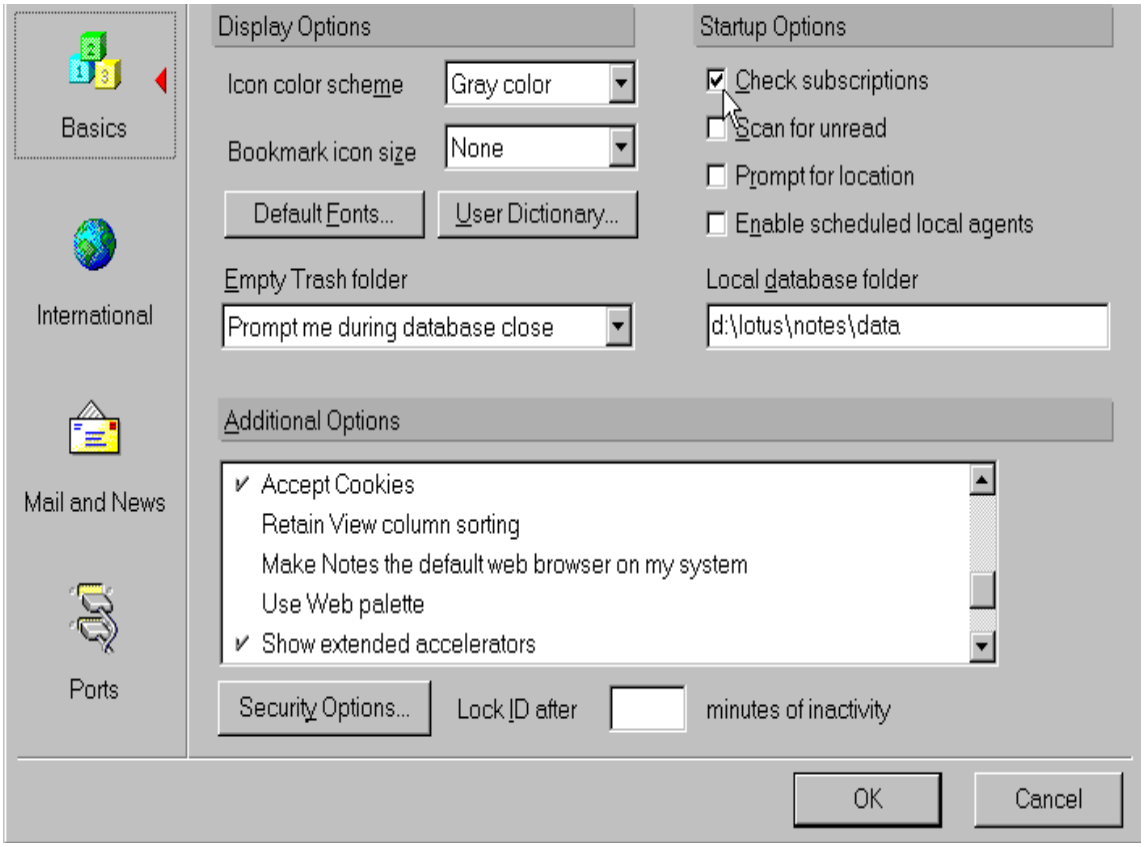


Figure 53. Checking for subscriptions in the user settings

Creating the subscriptions link

Once you have created the subscriptions database, follow these steps to make the corresponding link into the menu outline:

- Create a new outline entry and call it `Subscription` with alias `Subscription` and indent it underneath the Personal outline entry for the Notes client.
- Choose **Named Element** from the drop-down list in the outline entry box, and choose **Frameset** in the drop-down list to the right.
- Click the folder icon and the Locate Object dialog box opens.
- Click the drop-down list arrow in the Kind of object field and select **Frameset**.
- Click the drop-down list arrow in the Database field and select **Subscriptions**.

- Click the drop-down list arrow in the Frameset field and select **SubscriptionsFullscreen**.
- Click **OK** to close.
- Set the frame to **_blank** to open the subscriptions page in a separate window.
- Include an image if required.
- Remember to check the Web Browsers checkbox in the Hide outline entry from tab.
- Close the outline entry box and save the outline.

6.1.1.6 Sametime chat function

Sametime consists of client and server applications that enable a community of users to collaborate in real-time online meetings. Section 11.1, “Sametime” on page 217 describes more about Sametime, including the setup and integration of the Sametime chat function in the portal.

6.1.2 Web browser client

As we mentioned earlier in the book, we first created the personal mail, calendar, contacts, to do list, and subscriptions in a single page containing five frames in a frameset. We then split the page into single frames and added a link for each entry in the menu. This was more user-friendly and easier to use. But you may prefer a single page. Therefore, we first discuss building the applications into a single page containing frames and then continue with breaking the page up into different links.

We developed a special form named GenerateWebPersonalPage, which is used to support the frameset structure for the personal pages. This page is generated dynamically at the Web client during run-time using a combination of fixed HMTL code and a LotusScript agent named AgentGetMailUrl, which automatically extracts the personal information of the currently logged-in user.

We created the GenerateWebPersonalPage form with the following layout fields:

```
* Field name = User, Type = Names/Computed when composed, Value=@UserName

* Field name = MailUrl, Type = Text/Editable, Value=MailUrl

* Field name = Mail, Type = Text/Editable, Value=Mail

* Field name = Calendar, Type = Text/Editable, Value=Calendar

* Field name = ToDo, Type = Text/Editable, Value=ToDo
```

```
* Field name = Subscription, Type = Text/Editable, Value=Subscription
```

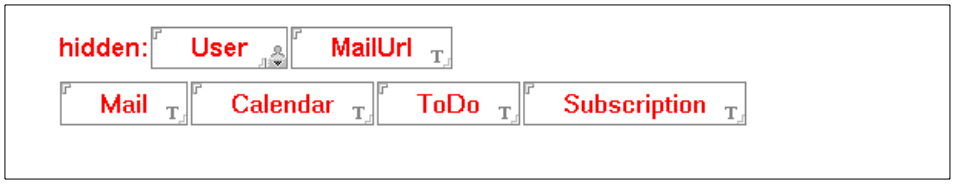


Figure 54. "GenerateWebPersonalPage" form layout

The User field is used to keep the full name of the current user. Based on this data we can get the information related to the currently logged-in user.

The contents of the other fields is generated by the AgentGetMailUrl agent, which is called at the WebQueryOpen form event using the formula:

```
@Command([ToolsRunMacro]; "AgentGetMailUrl")
```

This agent dynamically obtains the personal information of the current user that is logged into the portal. Once the content of the fields is completed, the information is used to complete the frameset structure created dynamically in the HTML Head Content form event, with the addition of the HTML code in Figure 55 on page 114.

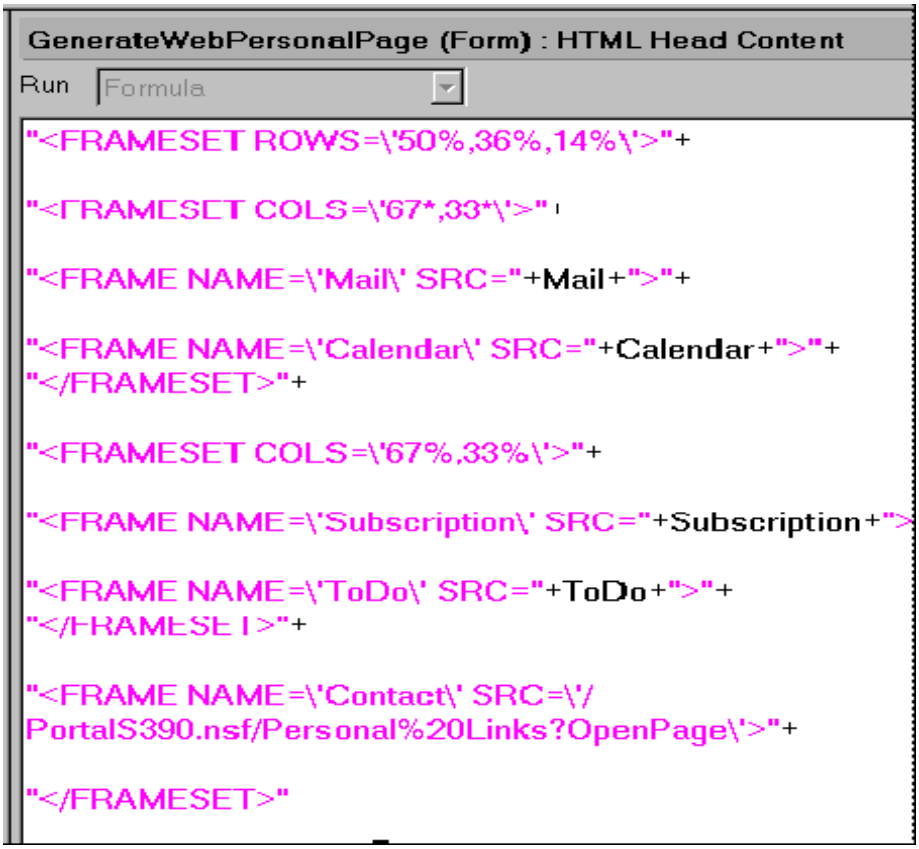


Figure 55. "FrameSet" layout from the personal portal page for Web environment

The process works in the following way:

- Once the form is invoked from the main menu, it executes the AgentGetMailUrl agent, which extracts the personal information of the currently logged-in user.
- The agent dynamically generates the links for the mail, calendar, to do list, subscriptions and contacts.
- The agent then put the links directly into the corresponding fields on the form and the form is displayed to the user.

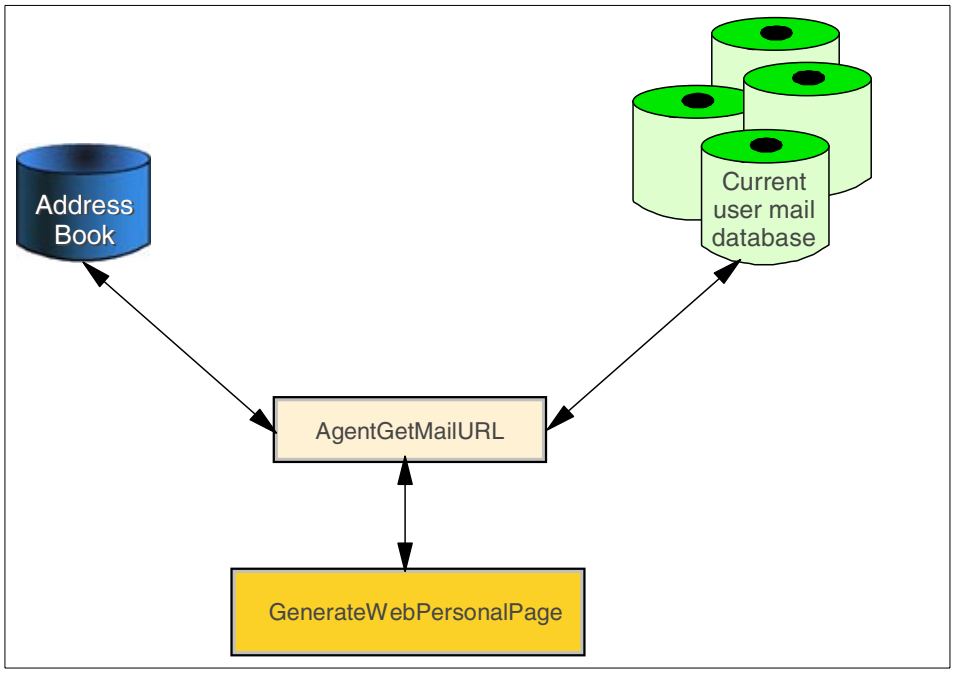


Figure 56. Generation of personal page frameset structure at Web client

In order to find which Domino Directory to use, we created the DomDir script library, and at the Option event we added:

```
Option Public
Option Declare
```

In the declaration event we added:

```
Const domdirpath$ = "names.nsf"
```

This code gave us the Domino Directory used by the “AgentGetMailUrl”.

With this we fix the Domino Directory used into the “AgentGetMailUrl” agent.

Following is the code of the “AgentGetMailUrl” LotusScript agent.

```
Sub Initialize
    Dim Session As New NotesSession
    Dim Db As NotesDatabase
    Dim DomDir As NotesDatabase
    Dim Doc As Notesdocument
    Dim PersDocColl As NotesDocumentcollection
    Dim PerDoc As NotesDocument
    Dim ServDocColl As NotesDocumentcollection
    Dim ServDoc As NotesDocument
```

```
    Dim agentLog As New NotesLog("Agent log")
    Call agentLog.OpenAgentLog

     'Get the full path of this database
    Call agentLog.LogAction("Get the full path of this database")
    Set Db = Session.CurrentDatabase
    Set DomDir = New Notesdatabase("",DomDirPath$)
    DbPath$ = Db.filepath

    Set doc = Session.DocumentContext
    NameUrl$ = Lcase$(doc.user(0))
    Call agentLog.LogAction("NameUrl$ = "+doc.user(0))
     'Search the mailbox via Persondocument entry
    SearchStr$ = "Form = ""Person"" & @lowercase(Fullname) = """ + NameUrl$ + """"

    Set PersDocColl = DomDir.Search(SearchStr$, Nothing, 0)

    'In case of multiple matches
    If PersDocColl.count > 1 Then   'multiple hierarchical name or rep. conflicts
        Call agentLog.LogAction("multiple hierarchical name or rep. conflicts")
        Print "<b>ERROR</b>The user has one or more conflicting names.<p>"
        Print "<b>Contact the Webmaster</b>"
        Print "<p><a href=" "OpenMail?OpenForm" " >Go back</a>"
        Exit Sub
    End If

    'Get Persondoc entries
    Call agentLog.LogAction("Get Persondoc entries")
    Set PersDoc = PersDocColl.GetFirstDocument
    Call agentLog.LogAction("Set PersDoc = PersDocColl.GetFirstDocument")
    If Not (PersDoc Is Nothing) Then

        'Replace \ with /
        Call agentLog.LogAction("Replace \ with /")
        Temp1$ = Trim(PersDoc.mailfile(0))
        pos = Instr(Temp1$,"\")
        Temp$ = Left$(Temp1$,pos-1) + "/" + Mid$(Temp1$,pos+1)

        If Instr(PersDoc.MailFile(0),".nsf") Then
            mail$ = Temp$
        Else
            mail$ = Temp$+ ".nsf"
        End If
    End If

    Server$ = PersDoc.MailServer(0)
    Call agentLog.LogAction("Server$ = PersDoc.MailServer(0)")

    SearchStr$ = "Form = ""Server"" & Servername = """ + Server$ + """"
    Set ServDocColl = DomDir.Search(SearchStr$, Nothing, 0)
    Set ServDoc = ServDocColl.GetFirstDocument
    If Not (ServDoc Is Nothing) Then
        Call agentLog.LogAction("For URL we use the NetAdr_0 field in the server
document")

        'For URL we use the NetAdr_0 field in the server document
        address$ = ServDoc.NetAddr_0(0)

        'Generate and assign the URL links to their corresponding fields

        doc.Mail ="'http://"+address$ + "/" + mail$ + "/($Inbox)?OpenView'"
        doc.Calendar = "'http://"+address$ + "/" + mail$ + "/($Calendar)?OpenView'"
        doc.ToDo = "'http://"+address$ + "/" + mail$ + "/($ToDo)?OpenView'"
```

```
        doc.Subscription =
"http://wtsc67oe/Subscrip.nsf/SubscriptionsFullscreen?OpenFrameset"
        Call doc.save(True,True)
        Exit Sub
    Else
        Call agentLog.LogAction("<b>ERROR</b>, mailfile or server not found.")
        Print "<b>ERROR</b>, mailfile or server not found."
        Print "<p><a href=""/" + dbpath$ + """>Go back</a> "
    End If
    Call agentLog.Close
End Sub
```

First, we get the full path of the portal database (current database) and then we obtain the full user's name. With this we make the search query to find the personal document for the current user in the admin.nsf database.

We then open the admin.nsf database and execute the search criteria. After the execution we verify the search's results. In the case where we get more than one personal document, we display a conflict error message indicating that the user has one or more conflicting names.

Once we get the personal document, we get the mail file of the current user and the name of the mail server. With this information we make the search query to find the server document for the appropriate server.

After obtaining the server document, we obtain the net address for the server. We then use this net address to generate the URLs for the personal elements of the current user. During the creation of the URLs, we assign them to the corresponding field of the "GenerateWebPersonalPage" document.

Using this process, we created a frameset containing all the personal elements. We then split the frameset into single links and added the links to the portal menu as described in the following sections.

Note

We added forms for each entry, containing a single frame for each application. This was the easiest option for us, since we already had the structure defined in the single form containing all the applications. This might not be the ideal way to do it if each link is opened separately.

6.1.2.1 Personal mail

The following steps describe the creation of the link in the menu outline:

- Create a form called `PersonalMail` with alias `PersonalMail`.
- Add the fields previously added to the GenerateWebPersonalPage form.
- Add the following to the HTML Head Content element of the form:

```
"<FRAMESET COLS=\'*\'>"+
"<FRAME NAME=\'Mail\' SRC="+Mail+">"+
"</FRAMESET>"
```

- Add the following to the WebQueryOpen element of the form:

```
@Command([ToolsRunMacro]; "AgentGetMailUrl")
```

- Save the form.
- Add an outline entry to the PortalMainMenu outline and indent it underneath the Personal entry for the browser client.
- Name this entry `Mail` with alias `Mail`.
- Choose **Named Element** from the drop-down list and choose **Form** in the drop-down list field to the right.
- Click the folder icon and the Locate Object dialog box opens.
- Click the drop-down list arrow in the Database field and select **-Current database-**
- Click the drop-down list arrow in the Form field and select **PersonalMail**.
- Click **OK** to close.
- Set the frame to **_blank** to open the mail page in a separate window.
- We also included an image for the menu entry, but this is optional and only for presentation purposes.
- Remember to check the Notes R4.6 or later checkbox in the Hide outline entry from tab.
- Close the outline entry box and save the outline.

6.1.2.2 Personal calendar

The following steps describe the creation of the link in the menu outline:

- Create a form called `PersonalCalendar` with alias `PersonalCalendar`.
- Add the fields previously added to the GenerateWebPersonalPage form.
- Add the following to the HTML Head Content element of the form:

```
"<FRAMESET COLS=\'*\'>"+
"<FRAME NAME=\'Calendar\' SRC="+Calendar+">"+
"</FRAMESET>"
```

- Add the following to the WebQueryOpen element of the form:

```
@Command([ToolsRunMacro]; "AgentGetMailUrl")
```

- Save the form.

- Add an outline entry to the PortalMainMenu outline and indent it underneath the Personal entry for the browser client.
- Name this entry `Calendar` with alias `Calendar`.
- Choose **Named Element** from the drop-down list and choose **Form** in the drop-down list field to the right.
- Click the folder icon and the Locate Object dialog box opens.
- Click the drop-down list arrow in the Database field and select **-Current database-**
- Click the drop-down list arrow in the Form field and select **PersonalCalendar**.
- Click **OK** to close.
- Set the frame to **_blank** to open the page in a separate window.
- Include an image for the menu entry if required.
- Remember to check the Notes R4.6 or later checkbox in the Hide outline entry from tab.
- Close the outline entry box and save the outline.

6.1.2.3 Personal contacts

The following steps describe the creation of the link in the menu outline:

- Create a form called `PersonalContacts` with alias `PersonalContacts`.
- Add the fields previously added to the GenerateWebPersonalPage form.
- Add the following to the HTML Head Content element of the form:

```
"<FRAMESET COLS=\'*\'>"+
"<FRAME NAME=\'Contacts\' SRC=\'/PortalS390.nsf/Personal%20Links?OpenPage\'>"+
"</FRAMESET>"
```

- Add the following to the WebQueryOpen element of the form:

```
@Command([ToolsRunMacro]; "AgentGetMailUrl")
```

- Save the form.
- Add an outline entry to the PortalMainMenu outline and indent it underneath the Personal entry for the browser client.
- Name this entry `Contacts` with alias `Contacts`.
- Choose **Named Element** from the drop-down list and choose **Form** in the drop-down list field to the right.
- Click the folder icon and the Locate Object dialog box opens.

- Click the drop-down list arrow in the Database field and select -**Current database-**.
- Click the drop-down list arrow in the Form field and select **PersonalContacts**.
- Click **OK** to close.
- Set the frame to **_blank** to open the page in a separate window.
- Include an image for the menu entry if required.
- Remember to check the Notes R4.6 or later checkbox in the Hide outline entry from tab.
- Close the outline entry box and save the outline.

6.1.2.4 Personal to do list

The following steps describe the creation of the link in the menu outline:

- Create a form called `PersonalToDo` with alias `PersonalToDo`.
- Add the fields previously added to the GenerateWebPersonalPage form.
- Add the following to the HTML Head Content element of the form:

```
"<FRAMESET COLS=\'*\'>"+
"<FRAME NAME=\'ToDo\' SRC="+ToDo+">"+
"</FRAMESET>"
```

- Add the following to the WebQueryOpen element of the form:

```
@Command([ToolsRunMacro]; "AgentGetMailUrl")
```

- Save the form.
- Add an outline entry to the PortalMainMenu outline and indent it underneath the Personal entry for the browser client.
- Name this entry `ToDo` with alias `ToDo`.
- Choose **Named Element** from the drop-down list and choose **Form** in the drop-down list field to the right.
- Click the folder icon and the Locate Object dialog box opens.
- Click the drop-down list arrow in the Database field and select **-Current database-**.
- Click the drop-down list arrow in the Form field and select **PersonalToDo**.
- Click **OK** to close.
- Set the frame to **_blank** to open the page in a separate window.
- Include an image for the menu entry if required.

- Remember to check the Notes R4.6 or later checkbox in the Hide outline entry from tab.
- Close the outline entry box and save the outline.

6.1.2.5 Personal subscriptions

As mentioned with the Notes client, in order to complete the subscriptions link, the subscription database has to be created and configured.

The following steps describe the creation of the link in the menu outline after the subscription database has been configured:

- Create a form called `PersonalSubscriptions` with alias `PersonalSubscriptions`.
- Add the fields previously added to the GenerateWebPersonalPage form.
- Add the following to the HTML Head Content element of the form:

```
"<FRAMESET COLS=\'*\'>"+
"<FRAME NAME=\'Subscriptions\' SRC="+Subscriptions+">"+
"</FRAMESET>"
```

- Add the following to the WebQueryOpen element of the form:

```
@Command([ToolsRunMacro]; "AgentGetMailUrl")
```

- Save the form.
- Add an outline entry to the PortalMainMenu outline and indent it underneath the Personal entry for the browser client.
- Name this entry `Subscription` with alias `Subscription`.
- Choose **Named Element** from the drop-down list and choose **Form** in the drop-down list field to the right.
- Click the folder icon and the Locate Object dialog box opens.
- Click the drop-down list arrow in the Database field and select **-Current database-**.
- Click the drop-down list arrow in the Form field and select **PersonalSubscriptions**.
- Click **OK** to close.
- Set the frame to **_blank** to open the page in a separate window.
- Include an image for the menu entry if required.
- Remember to check the Notes R4.6 or later checkbox in the Hide outline entry from tab.
- Close the outline entry box and save the outline.

6.1.2.6 Sametime chat function

11.1, “Sametime” on page 217 describes the Sametime application and also the integration of the Sametime chat function into the portal.

6.2 Enterprise portal pages

On the enterprise portal page we included some applications of general interest for the employees of a company. All company users have access to these pages.

We included the following applications in the enterprise portal:

- Domino domain search
- Quickplace
- Sametime
- News
- Newsgroups
- Discussion
- Library

6.2.1 Domino domain search

The purpose of the Domino domain search element is to have a search engine which can be used to search the whole site from a Notes client or a Web browser.

6.2.1.1 Introduction to the Domino domain search

Domain search allows users to search an entire Notes domain for documents, files, and attachments that match a search query. Domain search is centralized on a domain catalog server, which uses a domain catalog to control which databases and file systems to index. Domain search also filters results to users based on their access to the indexed databases and file systems.

The domain indexer builds a central domain index in the domain catalog server that all domain search queries run against. The domain catalog can replicate to other servers. The domain indexer generates the index files directly in the domain catalog server, and these files do not replicate to other servers.

Domain catalog
The domain catalog is a database that uses the CATALOG.NTF design. The catalog stores records indicating which databases and file systems the server should index and the forms used to search the index. By design, both the domain catalog and the domain index reside on the domain catalog server. Database designers and managers set a database to be indexed by enabling the database property “Include in multi database indexing.” Administrators can configure this setting for multiple databases using the Domino Administrator. Using the domain catalog, administrators specify which file systems to index.

Search from Notes
As soon as the designated domain catalog server is recorded in the users’ Location documents, Notes users can perform domain searches. For information on how Notes users execute a domain search, see Notes 5 Help.

Search from the Web
Web clients have access to most domain search functionality, including search queries and result sets.

6.2.1.2 Features of Domino domain search
Here we discuss some of the important features of Domino domain search.

Security
The domain catalog server uses the domain catalog to filter search results on Notes databases based on the user's level of access to the results. If a user does not have access to view a document, the server does not include it in the results.

Note

File system searches don’t work on a Domino S390 server.

Centralized index
There is no duplication of indexes. This saves disk space and ensures that users get up-to-date results.

Customize search and result forms
Application designers can customize forms to refine queries or limit user queries to specific databases. Search and result forms can include graphics, such as background images or corporate logos. The result form can be customized to display additional information about result documents, such as their author or when they were created.

Fuzzy search
In Domino R5, search returns results even if the query is misspelled; a search on “Montrael” would find results for “Montreal.”

Stemming
R5 search uses linguistic analysis to return documents that contain words related to the words in the search query; a search on “swim” returns results containing “swimming” and “swimmer.” For details on the domain search, refer to the Domino Administration and Client Help.

6.2.1.3 Setting up the domain search
Here are some tips on how to set up the domain search.

Notes client
For each Notes client, the name of the catalog server has to be in the location document used, as shown in Figure 57.

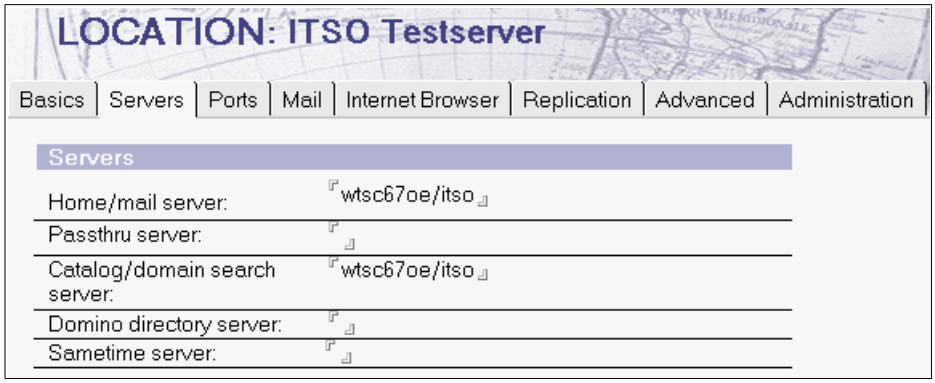


Figure 57. Setting up domain search on the Notes client

Domino server
A few steps must be followed in order to activate domain search on a Domino server. The following steps show how we set up domain search on our Domino for S/390 server.

1. The domain catalog has to be activated in the server document and the server executing the search has to be filled in, as illustrated in Figure 58 on page 125.

.

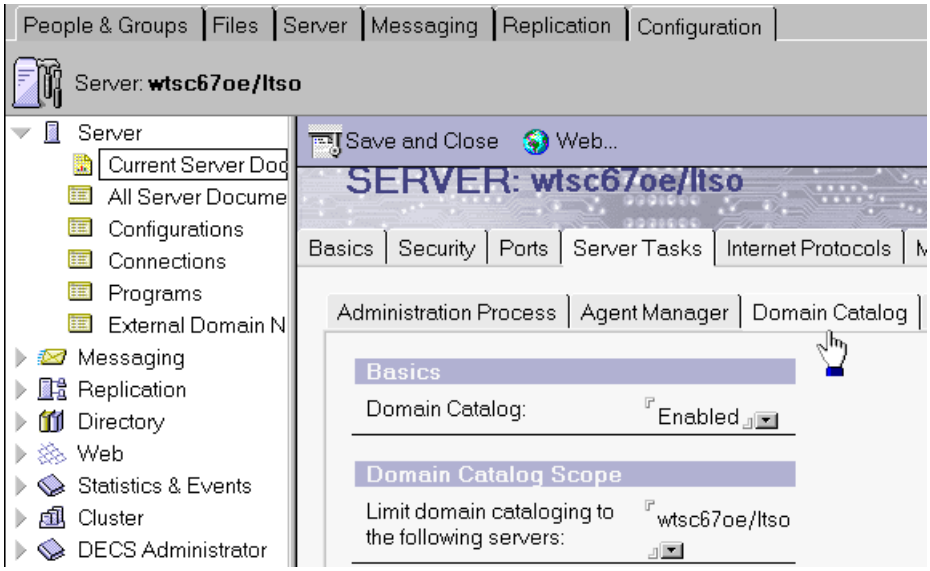


Figure 58. Activating Domino domain search

The timing of the indexing has to be set in the Server document under **Server Tasks -> Domain indexer**, as shown in Figure 59.

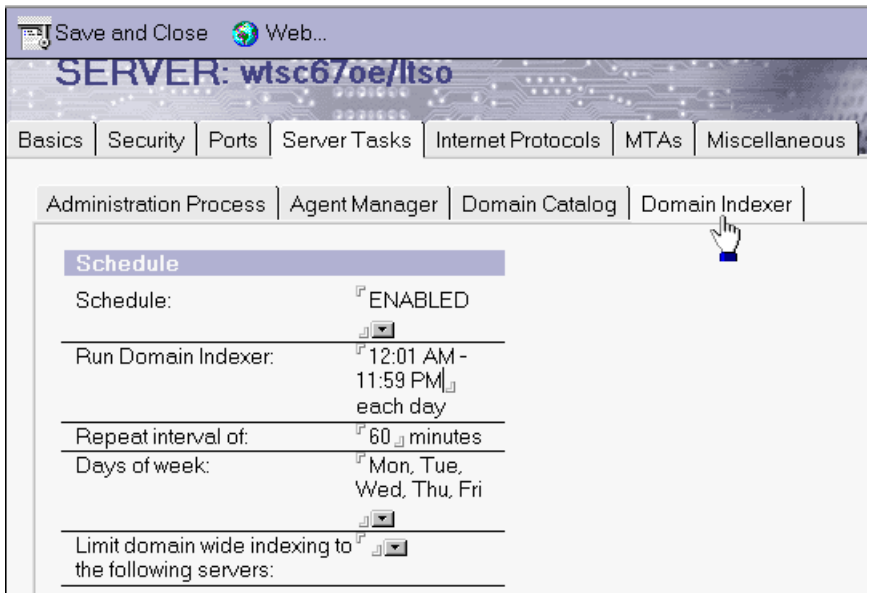


Figure 59. indexing time schedule for domain search indexer

2. Now the domain indexer task can be started. Before you start it, start the cataloger task to get the Domino catalog up-to-date by issuing `load catalog` on the Domino server console. You can start the domain indexer by issuing `load domidx` on the console. However, you should put it in the servertasks line in the Notes.ini or create a program document in the Domino Directory for this task.
3. For Web client domain search, fill in the host name in the Server Document -> Web Protocols tab, as shown in Figure 60.

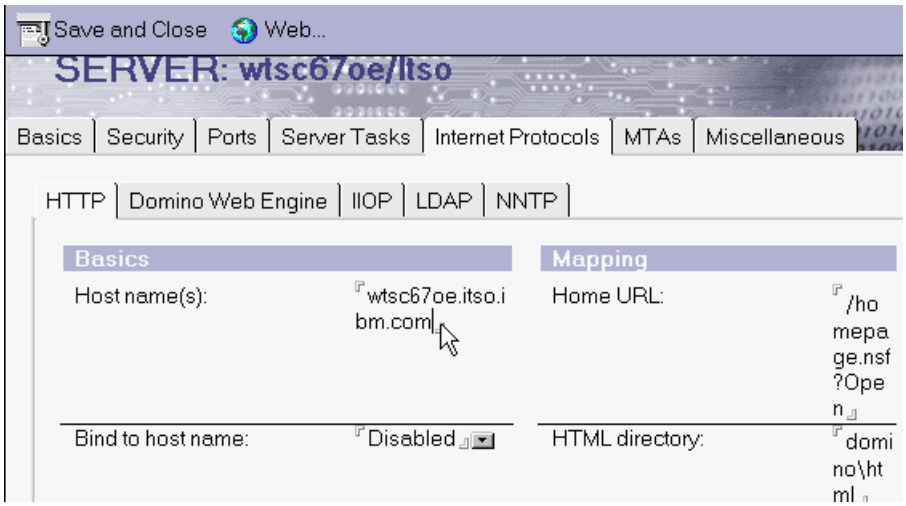


Figure 60. Entering the server's host name for Web client domain search

4. To add databases to the domain index, go to the Domino administrator, select the Files tab, select the database you want to include and select **Databases -> Multi database index**. In the pop-up window, select **Enable** and press **OK**. This is shown in Figure 61 on page 127.

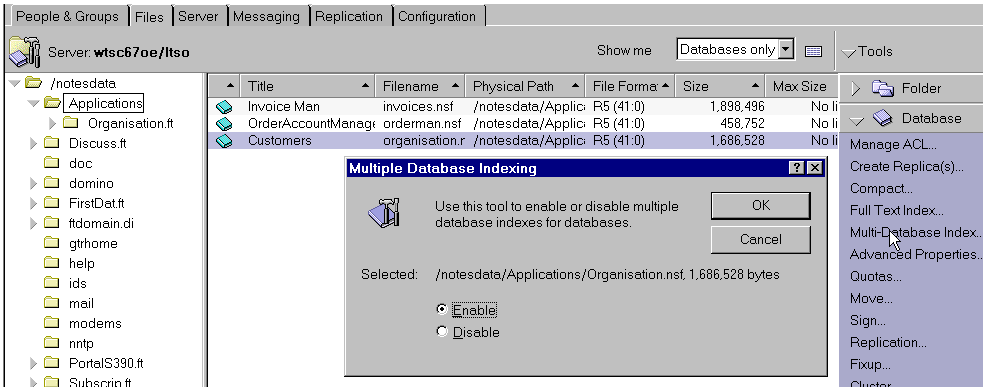


Figure 61. Adding databases to the domain search

5. Be sure to set the ACL correctly so the catalog server can access the database and documents, and the users who want to view the results from Domino or the Web have the correct access right on the documents.

6.2.1.4 Integrating a custom domain search interface

We created the custom search form in the catalog.nsf. To do that, we copied the domain query search form and renamed the new form to EnterpriseSearch.

We extracted only the SimpleQuery field and the three search buttons from the original table and copied them above the table. Some of the fields in the table are needed for proper operation so they cannot be deleted. We just hid them in Web and Notes versions.

We included a link to the enterprise discussion database and we added an Advanced search Hotspot which leads to the original catalog DomainQuery search form with all its options. This solution works in Notes and in the Web browser in exactly the same way. The form is illustrated in Figure 62 on page 128.

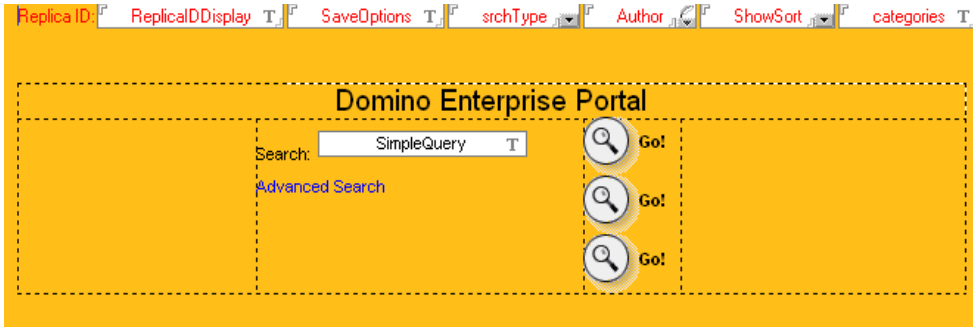


Figure 62. The domain search form in the enterprise portal

The form is added as a named element form to the header frame of the EnterprisePortal Frameset.

6.2.2 Quickplace

Quickplace is a self-service Web tool for team collaboration. We discuss it in detail in 11.2, “QuickPlace” on page 227.

6.2.3 Sametime

Sametime consists of client and server applications that enable a community of users to collaborate in real-time online meetings. Sametime is discussed in detail in 11.1, “Sametime” on page 217.

6.2.4 News

For our portal we included a page that contains news for the enterprise. We created a page that we called “EnterpriseNews” and added some news to the page. We then added an item in the portal menu, indented under the Enterprise entry, which opens the page named element EnterpriseNews.

In our case, one of the administrators kept the EnterpriseNews page up-to-date. In a real production environment, you can write an application to change the contents of the news page.

6.2.5 Newsgroups

Refer to Chapter 10, “NNTP and the Domino server” on page 197 for a complete description and instructions about incorporating newsgroups and NNTP into the portal.

6.2.6 Discussion database

6.2.6.1 What does this database do?

Almost any group that has information to share among its members can use a discussion database. A workgroup can use a discussion database to share their thoughts and ideas. An engineering group can discuss the products they are designing. An advertising agency can discuss the ad campaigns they are developing. A special interest group can share ideas and opinions on their common interests.

A discussion database is an informal meeting place, where the members of a workgroup can share ideas and comments. Like a physical meeting, each member of the workgroup listens to what others have to say and can voice his/her own opinions. However, unlike a physical meeting, the participants do not have to be in the same room at the same time to share information. People can participate when it is convenient for them to do so, and because it is easy for them to share information, they will do so.

To get started, a user can simply browse through discussion topics and responses that others have contributed. This is particularly useful for new workgroup members who want to learn about the issues that the group is working on. The history of discussion about these issues is preserved in the group's discussion database.

Users can also take a more active role in the discussion by composing their own responses to others' comments and by proposing new main topics for discussion.

6.2.6.2 Who will use this database?

All the members of a workgroup should use the group's discussion database. Some users will just follow the main topics and responses that other members have contributed. More active users will compose their own main topics and responses.

This template is designed for users accessing the information either from a Notes client or a Web browser. Most features are available to both types of users. See the *Using Database* document for more information on how to use the discussion features from a Web browser. To access the Using Database document:

1. Open the discussion database with the Lotus Domino Designer client.
2. Click the **Resources** folder and expand its options.
3. Click **Other**.
4. Double-click **Using Database**.

6.2.6.3 When do you use this database?

Use the database when you have time: a free moment between meetings, after hours, or while traveling.

6.2.6.4 Where can you find more information?

More information can be found in the R5 help database, HELP5.NSF and in the *Using this database* document for this database. Refer to the documentation database at:

```
http://www.notes.net Best Practices: Templates and Sample
           Databases(BPTEMP.NSF)
```

6.2.6.5 Access control

The default access of a database created with this template is Author.

6.2.6.6 Adding a link in the portal for the database

For the Notes client we added an entry in the portal menu outline and indented it under the Enterprise entry. The entry was named `Discussion` with alias `Discussion` and contains the following computed action:

```
@Command([FileOpenDatabase]; "wtsc67oe" : "Discuss.nsf" )
```

This will open the discussion database in the Notes client environment without any other customizing needed.

For the browser environment, we added an entry in the menu outline and indented it under the Enterprise entry. The entry was named `Discussion` with alias `Discussion` , and contains the following URL:

```
http://wtsc67oe.itso.ibm.com/Discuss.nsf/MasterDiscFrameset?OpenFrameset
(wtsc67oe.itso.ibm.com is our server domain name)
```

This will open the discussion database in the browser environment.

Note: Remember to hide the Notes entry from browser environments and the browser entry from the Notes client environment.

6.2.7 Library

6.2.7.1 What does this database do?

A document library application is an electronic filing cabinet that stores reference documents for access by a workgroup. The database might contain anything from environmental impact statements for a group of engineers to financial statements for a group of loan officers.

6.2.7.2 Who will use this database?

Anyone who wishes to create a record of a document or review available documents may use this database.

6.2.7.3 Important Features

- The database can be accessed from either a Web browser or a Notes client.
- The database can be used to route a document to a series of recipients during a review cycle.
- Document archiving can be achieved by moving expired documents to an archive database.

6.2.7.4 Suggestions for modifications

Full text search: If you wish to use the Notes full text search capabilities for a database created with this template, select **File -> Database -> Properties**, go to the Full Text panel, and then select **Create Index...** to create the full text index.

6.2.7.5 Where to find more information?

More information can be found in the *Using This Database* document that is part of Notes online help, or refer to the documentation database at:

```
http://www.notes.net: Best Practices: Templates and Sample
              Databases (BPTEMP.NSF).
```

6.2.7.6 Access Control

It is important to note that the access level should be *Author* for all users of this database. This will prevent unauthorized editing of documents within the database. The Author fields within the forms govern who will be able to edit/review particular documents. Errors will occur if someone with Editor access attempts to review a document when they are not an authorized reviewer of that document. For those accessing the database from a Web browser, the database does not accept Anonymous users.

6.2.7.7 Redesigned Document Library 4.0 Databases

Sometimes users upgrade the design of their databases using a template provided in a new release of Notes. There is a problem with Doclib4.ntf databases which are redesigned using the Doclbw50.ntf template (users are unable to open the documents in the database).

Doclib4 users: After you replace/refresh the design of your database using the Doclbw50 template, select **Actions -> Upgrade Doclib R4 Documents**

to New Design. This will run on all documents that were created with the R4 design, and upgrade them to the R4.6 design.

If you have created the Document Library using either the Doclbw46.ntf (Std46WebDocLib) or Doclbw50.ntf (StdR50WebDocLib) template, there is no need to use this agent.

6.2.7.8 Adding a link in the portal for the database

For the Notes client we added an entry in the portal menu outline and indented it under the Enterprise entry. The entry was named `Library` with alias `Library` and contains the following computed action:

```
@Command([FileOpenDatabase]; "wtsc67oe" : "DocLibra.nsf" )
```

This will open the library database in the Notes client environment.

For the browser environment, we added an entry in the menu outline and indented it under the Enterprise entry. The entry was named `Library` with alias `Library` and contains the following URL:

```
http://wtsc67oe.itso.ibm.com/DocLibra.nsf?OpenDatabase
(wtsc67oe.itso.ibm.com is our server domain name)
```

This will open the library database in the browser environment.

Note: Remember to hide the Notes entry from browser environments and the browser entry from the Notes client environment.

Chapter 7. Adding functional and administration pages

In this chapter we continue to add applications to the portal. These include the functional pages and the administration pages.

7.1 Functional portal pages

We include two examples of functional pages in the portal application: the sales page and the technical support page.

7.1.1 Sales page

This page is exclusively for users with the [Sales] role assigned in the portal application, so this page shows applications of particular interest to the sales department.

We include the following applications in this page:

- TeamRoom
- Customer

7.1.1.1 TeamRoom

TeamRoom is a Lotus Notes application designed to support processes that help people work together. The software does not create the sense of shared purpose, common language, focus, and drive which make effective teams productive. But, by embedding in its technology a template for many of the basic practices of good teaming, TeamRoom facilitates the creation and ongoing development of processes and practices common to high performance teams.

TeamRoom is a tool for information sharing and collaboration. However, the work of a TeamRoom-based team is not fundamentally different than it was before groupware. By creating a shared context for team work, TeamRoom does what many good tools do: creates leverage, both for the individuals on the team and for the team as a whole. Because the technology is built on a Notes groupware “sharing” platform, this context is richer and the leverage is greater than is typically possible with “sending” tools like e-mail. Lotus Notes also makes it possible for TeamRoom to support teams whose members are geographically distributed, and who are sometimes unable to be connected to their computer network.

Examples of work made easier and more efficient by TeamRoom include:

- Raising and discussing issues and concerns.

© Copyright IBM Corp. 2001

- Creating (collaborative) products: memos, presentations, other deliverables.
- Brainstorming (leading to resolution and action).
- Preparing for meetings: team members can present and share information in the TeamRoom before the meeting, so that meeting time can be focused on decision making.
- Tracking meeting agendas and resulting action items.

7.1.1.2 Customer

The Customer application supports a simple sales process, a service the company offers to its customers. They are able to input requirements about products the company offers, and get direct access to the product catalog in real time. The application could be accessed using either a Web browser or Notes client.

The Customer application is a hybrid, composed of a front-end developed with Lotus Notes and a back-end developed with DB2. Both environments in communication with each other using DECS, a feature of Domino. See details at 8.2.3, “The sample portal customer application” on page 165.

7.1.2 Tech support page

Only the users with the [TechSupport] role assigned have access to this page and the applications that are part of it.

We include the following applications in this page:

- TeamRoom
- Part order

7.2 Administration pages

Once we have constructed the personal, functional, and enterprise pages, we are ready to include the content of the main portal page. We have a frameset composed of two frames: the left frame is allocated to our navigation mechanism and the right frame is able to display the content of the current page we have selected.

7.2.1 Portal configuration

This is a very useful element in the application. With this we are able to make parameters for a variety of special elements that can be changed or adapted according to the business needs.

The configuration interface is accessed from the main menu of the portal application. It can be accessed only by users with the [PortalAdmin] role assigned. It is composed of Consult and New options.

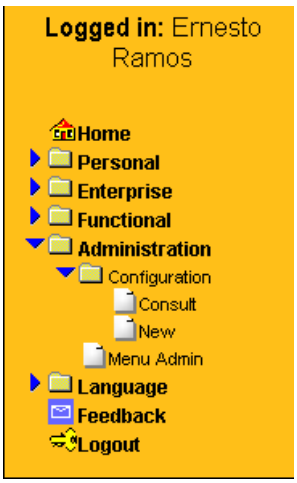


Figure 63. Configuration choice

When the user clicks this option, the application displays the screen to create a new parameter document; see Figure 63. Developers are able to create the parameters they consider necessary for the portal application. This option could also be used by an administrator if they need to re-create a parameter document that is accidentally removed from the application.

7.2.1.1 New option

Here the user must fill out the concept and content fields. In the Concept field, the user puts a key word to identify the parameter that must be used in the application. The Content field records the value of the parameter used in the application.

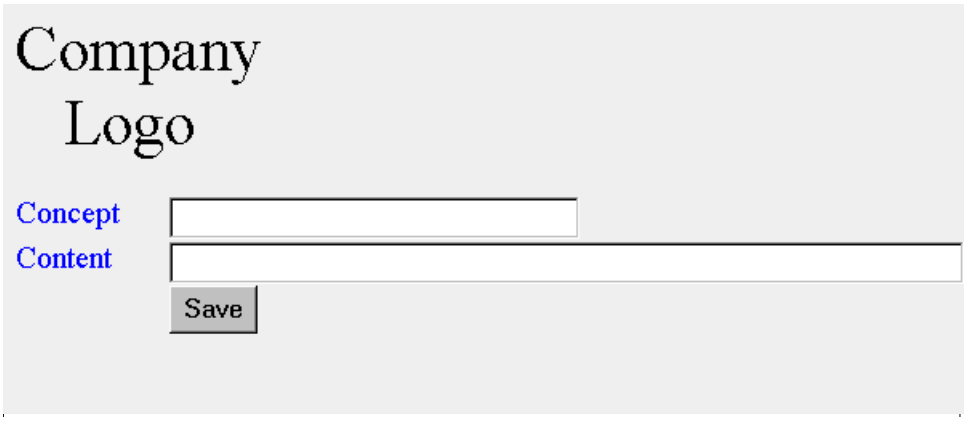


Figure 64. New parameter interface

7.2.1.2 Consult option

When the user clicks the Consult option, the application displays a screen with a view of all the parameters. If the user wants to modify any parameter, he can simply double-click it, and the application will display the format with the data able to be modified.

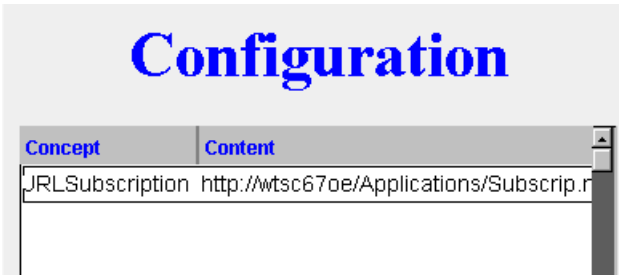


Figure 65. Configuration view for consults

7.2.1.3 Build interface

The configuration interface is composed of the following design elements:

- Configuration form
- ViewConfiguration view
- ConfigurationPage page

The configuration form is composed of the following elements:

```
* Field name = Concept, Type = Text/Editable, Value=Concept
* Field name = Content, Type = Text/Editable, Value=Content
```

```
* Button name = Save, Formula = @Command([FileSave])
```

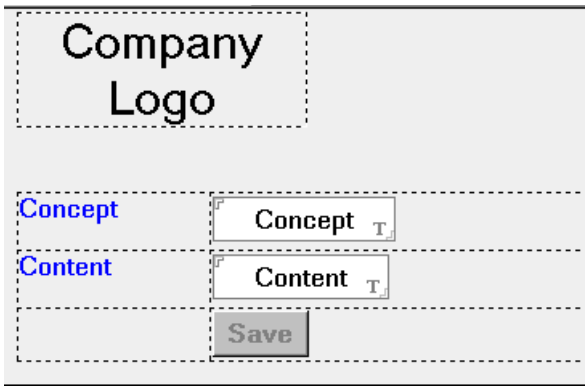


Figure 66. Configuration form layout structure

Create the ViewConfiguration view as shown in Figure 67.

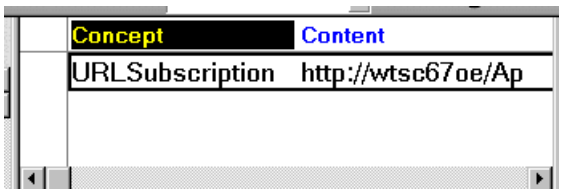


Figure 67. ViewConfiguration view layout structure

The view formula is:

```
SELECT form = "Configuration"
```

Create the following columns:

1. `Column name = Concept, Sort = Ascending, Type = Standard, Formula = Concept`
2. `Column name = Content, Sort = None, Type = Standard, Formula = Content`

The ConfigurationPage page is composed of one element, the embedded view ConfigurationView. This is shown in Figure 68.

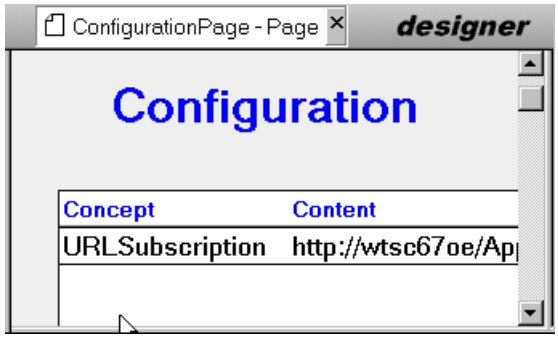


Figure 68. ConfigurationPage page layout structure

7.2.2 Menu generator

This is a special interface built to administer the choices for all the menus for the different functional pages included in our portal application. This interface is only available to users with the [PortalAdmin] role, and it can be accessed through the administration choice on the navigation menu of the portal application.

Through this interface we can add or erase choices for one particular option's menu (from a specific functional page), and we don't need to use the Notes design client to complete that activity. Portal administrators must be able to add, update, or remove choices as necessary, without having to rely on assistance from specialized technical developers.

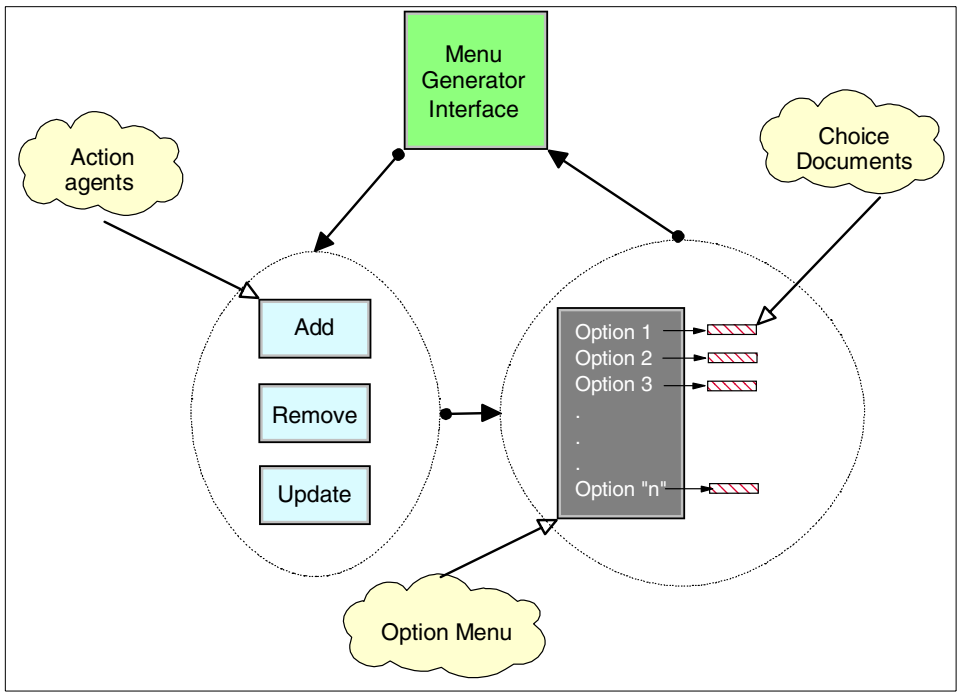


Figure 69. Menu Generator components

The main element of the menu generator is the form FunctionalMenus. It is the interface used for administration of the functional page options menus.

To access this interface select **Administration** from the navigation area on the main page, and then click **MenuAdmin**. The right-hand frame contains the screen showing the Options Menu Administration interface, as shown in Figure 70 on page 140.

.

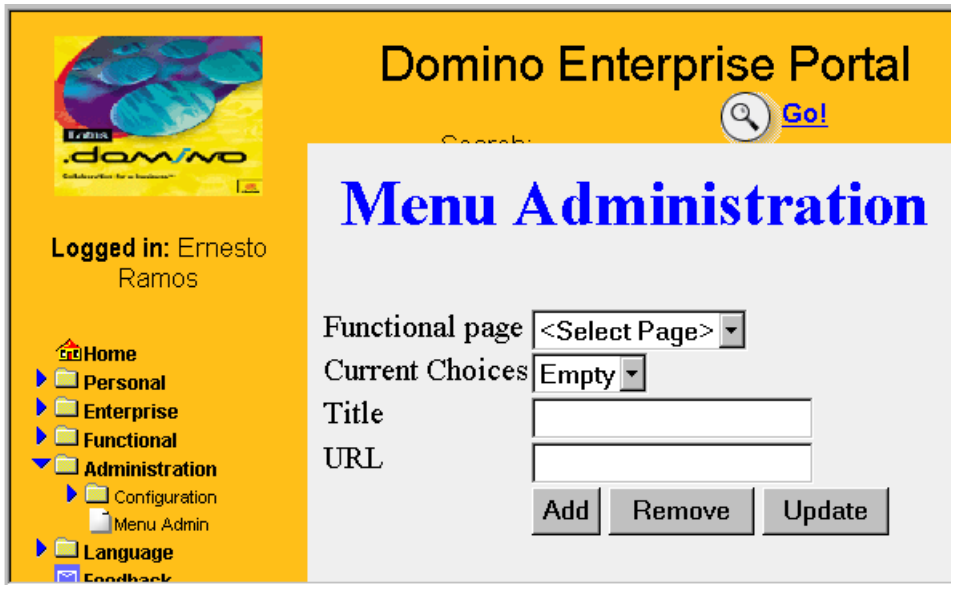


Figure 70. Options Menu Administration interface

The interface includes the following fields:

- Functional page: This field is used to select the functional page in which we want to work. In our portal we can select Sales or TechSupport.
- Current choices: This combobox displays the list of choices included on the Options menu from the selected functional page.
- Title: This is the title that appears on the Choices menu of the functional page.
- URL: This field is used to record the URL for the application we want to include in the functional page.

The buttons on the interface execute the following actions:

- Add: Executes the agent that generates the new entry for the menu of the selected functional page.
- Remove: Erases the selected choice on the current choices field for the selected functional page.
- Update: Updates any specific choice in a particular option menu.

To create a new choice in a particular option menu from a specific functional page, simply select the corresponding functional page from the Functional page drop-down list. This automatically fills the Current Choices field with the

corresponding choices available on the options menu from the selected functional page. Record the title of the choice in the Title field, and record the URL of the application you want to include in the URL field. Then click **Add**. The new entry is aggregated automatically into the choices menu of the corresponding functional page; you will see that it is now included in the Current Choices field. Later, when we navigate to the functional page, the choice will be ready for use.

To remove a choice from an option menu on a functional page, select the functional page in the Functional page field. Select the desired item from the Current Choices field and click **Remove**. The specified item is removed from the Current Choices field; when you access the functional page the choice does not appear in the menu.

7.2.2.1 The components of the menu generator

The menu generator is an extra infrastructure application. It is very helpful in administering all the option menus for the portal application. This interface is composed of many design elements that are hidden for the user. In this section we explain how to build the screen interface for the menu generator, then we build and integrate other design elements involved in the solution. Finally, we show how we built the code for our LotusScript agents. We use these to add and remove the different options in the functional page menus included in our portal application.

FunctionalMenus form

The form used to create functional menu items is shown in Figure 71.

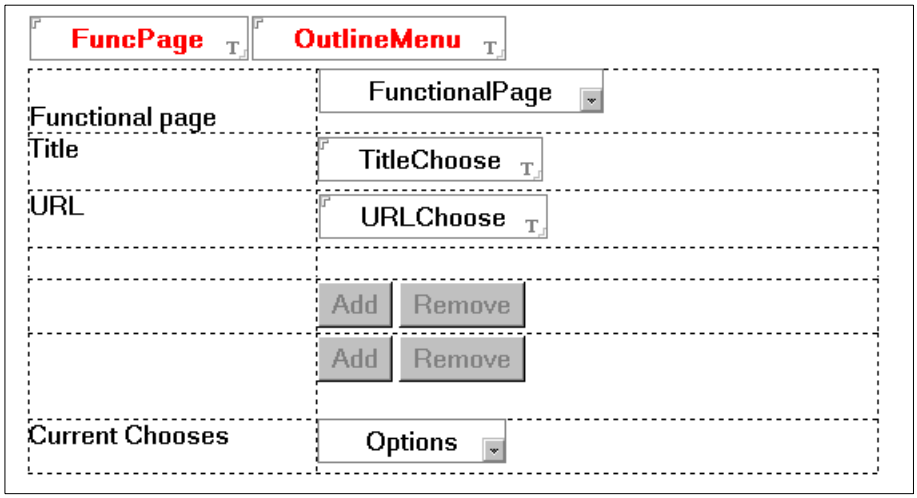


Figure 71. FunctionalMenus form design structure

First we have to create the following hidden fields:

```
*Field name = FuncPage, Type =Text/Computed,
Value=@DbColumn("":"NoCache";"":"";"ViewFuncPage";1)

*Field name = OutlineMenu, Type =Text/Computed,
Value=@DbColumn("":"NoCache";"":"";"ViewFuncPage";2)
```

These fields retrieve the related list of the functional pages available at the portal application, with their corresponding options menus (outline menus). The information from these fields is used in the LotusScript agents to make the match between the functional page and its options menu. LotusScript agents are discussed in more detail later in this section.

At this time, we incorporate the view ViewFuncPage and the form FunctionalPage. Both elements integrate the catalog of functional pages with their associated option menus.

Create the FunctionalPage form shown in Figure 72.

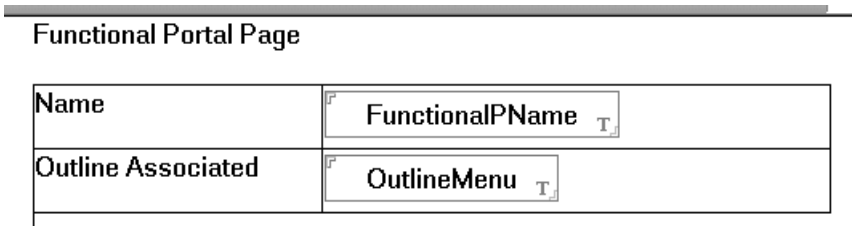


Figure 72. FunctionalPage form design structure

Create the following fields:

```
*Field name = FunctionalPName, Type =Text/Editable.
*Field name = OutlineMenu, Type =Text/Editable.
```

The FunctionalPName field is used to record the name of the functional portal, and the OutlineMenu is used to record the name of the outline choices menu included at the functional page.

Create the ViewFuncPage shown in Figure 73.

Figure 73. ViewFuncPage view design structure

The view formula is:

```
SELECT form= "FunctionalPage"
```

Create the following columns:

1. Column name = Functional Page, Sort = Ascending, Type = Standard, Formula =FunctionalPName.
2. Column name = Outline Menu, Sort = None, Type = Standard, Formula = OutlineMenu.

This view is used to manipulate the documents that record the relationship between the functional page and its outline menu.

Getting back to the FunctionalMenus form, we need to create the following of fields that work in a related way:

1. Field Name = FunctionalPage, Type = Combobox/Editable, Choises: "Use formula for choises"/Formula ="<Select Page>":@DbColumn("":"NoCache";"":"";"ViewFuncPage";1) Options: "Refresh fields on keyword change"
2. Field Name = Options, Type = Combobox/Editable, Choises: "Use formula for choises"/Formula="@If(@IsError(@DbLookup("":"NoCache";"":"";"ViewFuncChoose";@Text(FunctionalPage);2));"Empty";@DbLookup("":"NoCache";"":"";"ViewFuncChoose";@Text(FunctionalPage);2))

The FunctionalPage field is filled out with the list of the functional pages available on the portal. The FuncPage hidden field is filled out in the same way; by default it displays the <Select Page> message.

Once the functional page is selected, the formula for the Options field for the selected functional page is automatically recalculated to display the appropriate choices.

In the formula for the Options field, we call the view ViewFuncChoose, which is related to the Functionalchoices form. Both elements integrate the catalog of functional pages that are associated with their menu's choices. The Functionalchoices form is an auxiliary element which produces the mapping with the choices included on the menus of the functional pages. So they are generated and removed in the Lotus Script agents to add or remove choices on the functional page menus.

Create the Functionalchoices form shown in Figure 74 on page 144.

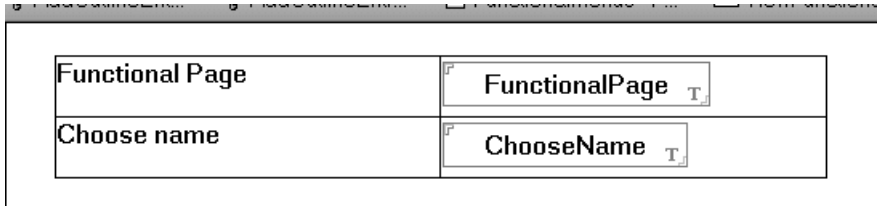


Figure 74. Functionalchoices form design structure

Create the following fields:

1. Field name = FunctionalPage, Type = Text/Computed, Value = FunctionalPage.
2. Field name = ChooseName, Type = Text/Computed, Value = ChooseName.

The FunctionalPage field is used to record the name of the functional page, and the ChooseName field is used to record the name of the choice that appears in the choices menu of the functional page.

Create the ViewFuncChoose view shown in Figure 75.

Functional Page	Functional Choose
Sales	Discussion 2
Sales	Discussion 3
Sales	TeamRoom
Sales	Armelle
Sales	Test

Figure 75. "ViewFuncChoose" view design structure

The view formula is:

```
SELECT form = "Functionalchoices"
```

Create the following columns:

```
1) Column name = Functional Page, Sort = Ascending, Type = Standard,
Formula =FunctionalPage
2) Column name = Functional Choose, Sort = None, Type = Standard,
Formula = ChooseName
```

This view is used to manipulate the documents with the relation between the functional page and all choices included in its choices menu.

For the FunctionalMenus form, we create the following elements:

```
1. Field name = TitleChoose, type = Text/Editable, DefaultValue =
   TitleChoose.
2. Field name = URLChoose, type = Text/Editable, DefaultValue = URLChoose.
3. Button label = Add, Hide pragraph form: "Notes R4.6 or later"
4. Click = @Command([ToolsRunMacro]; "AddOutlineEntryWeb")
5. Button label = Add, Hide pragraph form: "Web browsers"
6. Click = @Command([ToolsRunMacro]; "AddOutlineEntryNotes")
7. Button label = Remove, Hide pragraph form: "Notes R4.6 or later"
8. Click = @Command([ToolsRunMacro]; "RemoveOutlineEntryWeb")
9. Button label = Remove, Hide pragraph form: "Web browsers"
10.Click = @Command([ToolsRunMacro]; "RemoveOutlineEntryNotes")
```

The TitleChoose field is used to record the title of the new choice we are integrating to the functional page choices menu. The URLChoose field records the link to the application we are referring to in the TitleChoose.

We have two Add buttons. Both are used to add the new choice to the selected functional page. The difference is that one executes the AddOutlineEntryWeb LotusScript agent, which was designed to work in a Web environment, while the other executes the AddOutlineEntryNotes LotusScript agent, which was designed to work in a Notes client environment.

Likewise, we have two Remove buttons. Both are used to remove a specific choice from the choices menu of the functional page; one executes the RemoveOutlineEntryWeb LotusScript agent, the other executes the RemoveOutlineEntryNotes LotusScript agent.

To continue, we explain the functionality of both AddOutlineEntryWeb and RemoveOutlineEntryWeb agents. The code for the Notes client agent is in A.4, “The menu generator interface” on page 249.

AddOutlineEntryWeb agent

When we created this agent, we focused our attention on the logic and functionality of the process. If you want to use this agent sample, you will need to add code to validate your data and to implement more security.

First of all, we declare the objects required to manipulate the current database (db), the current document (doc), the document that represents the new choice to be included (doc2), the outline to manipulate the options menu for the functional page (outline), the NewEntry to add a new option to the menu (NewEntry), and a NotesLog object to make the tracking of the agent execution.

```
Sub Initialize
    Dim session As New NotesSession
    Dim db As NotesDatabase
    Dim doc As NotesDocument
    Dim doc2 As NotesDocument
    Dim outline As NotesOutline
    Dim NewEntry As NotesOutlineEntry
    Dim agentLog As New NotesLog("Agent log")

    Call agentLog.OpenAgentLog
    Call agentLog.LogAction("AddOutlineEntry")
```

After that, we get the current database and the current document. With this, we get the name of the functional page required, the title of the choice, the URL of the choice, and the hidden fields with the list of all functional pages and their corresponding option menus.

```
    Set db = session.CurrentDatabase
    Set doc = Session.DocumentContext
    Call agentLog.LogAction("Before read parameters")
    'Get the parameters data
    FunctionalPage = doc.FunctionalPage(0)
    TitleChoose = doc.TitleChoose(0)
    URLChoose = doc.URLChoose(0)
    FuncPage = doc.FuncPage
    OutlineMenu = doc.OutlineMenu
    Call agentLog.LogAction("After read")
```

The next step is to get the options menu for the functional page selected. But before that, we need to get the name of the outline design element that is supporting the required options menu. So we write a routine that searches for this element in the hidden fields that store the list of the functional pages and their outline design elements.

```
    'Get the corresponding Outline Menu
    NumElements = Ubound(FuncPage)
    For i = 0 To NumElements
        If ( Cstr(FunctionalPage) = Cstr(FuncPage(i)) )Then
            Menu = OutlineMenu(i)
            Set outline = db.GetOutline(Cstr(Menu))
            FrameName = "Body"
            Call agentLog.LogAction(Cstr(Menu))
            Exit For
        End If
    Next
```

We compare the functional page that is in the list with the functional page specified for the user. If they are the same, we get the name of the outline menu and use this name as a parameter to retrieve the outline required, with the method GetOutline from our database object.

```
Set outline = db.GetOutline(Cstr(Menu))
```

By the way, we get the name of the frame in which we want to display the application we launch through the choice we are adding.

Once we have the required outline design element, we create the new entry using the method CreateEntry from the outline object, and at the same time we set the title of this element, its URL (SetURL) and the frame (FrameText) in which this element must be displayed. At the end we save the changes to the outline element.

```
    Set NewEntry = outline.Createentry(Cstr(TitleChoose))

    If ( NewEntry.SetURL(Cstr(URLChoose)) Then
        NewEntry.FrameText = Cstr(FrameName)
    Else
        Call agentLog.LogAction("URL not assigned")
    End If

    Call outline.save()
    Call agentLog.LogAction("New entry generated")
```

Once we create the new entry, we must generate its corresponding “functional choices” document. This is displayed in the administration interface, to consult it or to remove it from its corresponding functional page. We set the functional page and the title of the recently generated entry.

```
    'Generate the corresponding functional choose document for administration proposals
    Set doc2 = db.CreateDocument
    doc2.Form  = "Functionalchoices"
    doc2.FunctionalPage = Cstr(FunctionalPage)
    doc2.ChooseName = Cstr(TitleChoose)
    Call doc2.Save( True, True )

    Call agentLog.LogAction("functional choose document already generated")
    Call agentLog.Close

End Sub
```

RemoveOutlineEntryWeb agent

This agent manipulates the same elements that were manipulated in the AddOutlineEntryWeb agent, and we add the view object.

```
Sub Initialize
    Dim session As New NotesSession
    Dim db As NotesDatabase
    Dim doc As NotesDocument
    Dim doc2 As NotesDocument
    Dim outline As NotesOutline
    Dim entry As NotesOutlineEntry
    Dim View As NotesView

    Dim agentLog As New NotesLog("Agent log")
    Call agentLog.OpenAgentLog
    Call agentLog.LogAction("Begin RemoveOutlineEntryWeb")
```

We get the current database, the current document, the parameters corresponding to functional page, and the name of the option selected.

```
Set db = session.CurrentDatabase
Set doc = Session.DocumentContext

'Get the parameters data
Call agentLog.LogAction("Before read parameters")
FunctionalPage = doc.FunctionalPage(0)
Options = doc.Options(0)
FuncPage = doc.FuncPage
OutlineMenu = doc.OutlineMenu
Call agentLog.LogAction("After read parameters")
```

Also, as we did for the AddOutlineEntryWeb agent, we locate the corresponding outline design element that supports our options menu.

```
'Get the corresponding Outline Menu
NumElements = Ubound(FuncPage)
For i = 0 To NumElements
    If ( Cstr(FunctionalPage) = Cstr(FuncPage(i)) )Then
        Menu = OutlineMenu(i)
        Set outline = db.GetOutline(Cstr(Menu))
        Exit For
    End If
Next
```

After that, we proceed to locate the entry we want to remove from the outline design element (options menu). To do that, we compare the “label” of each entry in the outline element with the “Options“ selected by the user.

```
'Get the required entry
Set entry = outline.GetFirst()

If Not(entry Is Nothing) Then

    Do

        If ( Cstr(entry.Label) = Cstr(Options) ) Then

            Call outline.RemoveEntry(entry)
            Call agentLog.LogAction("OutlineEntry Removed")

            ' Get the functional choose doc
            Set view = db.GetView("ViewFuncChoose")
            Set doc2 = view.GetFirstDocument

            If Not(doc2 Is Nothing) Then

                Do
                    If (  (Cstr(doc2.FunctionalPage(0)) = Cstr(FunctionalPage)) And
(Cstr(doc2.ChooseName(0)) = Cstr(Options)) )Then
                        Call doc2.Remove(True)
                        Call agentLog.LogAction("Functional choose removed")
                        Exit Do
                    End If
                    Set doc2 = view.GetNextDocument(doc2)
```

```
                Loop While ( Not(doc2 Is Nothing) )

            End If

            Exit Do

        End If

        Set entry = outline.GetNext(entry)

    Loop While Not(entry Is Nothing)

End If

Call outline.save()

Call agentLog.LogAction("End RemoveOutlineEntryWeb")
Call agentLog.Close

End Sub
```

Once we have found it, we apply the RemoveEntry method from the outline object. This removes our entry from the outline element:

```
Call outline.RemoveEntry(entry)
```

Then we locate the associated FunctionalChoose document to remove it from the list in the administration interface. Finally, we save the modifications of the outline options menu.

UpdateOutlineEntryWeb agent

The UpdateOutlineEntryWeb agent combines the logic of both the AddOutlineEntryWeb and RemoveOutlineEntryWeb agents. So, we declare and manipulate the same elements that were manipulated by the other agents.

```
Sub Initialize
    Dim session As New NotesSession
    Dim db As NotesDatabase
    Dim doc As NotesDocument
    Dim doc2 As NotesDocument
    Dim outline As NotesOutline
    Dim entry As NotesOutlineEntry
    Dim View As NotesView

    Dim agentLog As New NotesLog("Agent log")
    Call agentLog.OpenAgentLog
    Call agentLog.LogAction("Begin UpdateOutlineEntryWeb")

    Set db = session.CurrentDatabase
    Set doc = Session.DocumentContext

    Call agentLog.LogAction("Before read parameters")
    'Get the parameters data
    FunctionalPage = doc.FunctionalPage(0)
    Call agentLog.LogAction(Cstr(FunctionalPage))
```

```
Options = doc.Options(0)
Call agentLog.LogAction(Cstr(Options))
TitleChoose = doc.TitleChoose(0)
Call agentLog.LogAction(Cstr(TitleChoose))
URLChoose = doc.URLChoose(0)
Call agentLog.LogAction(Cstr(URLChoose))
FuncPage = doc.FuncPage
OutlineMenu = doc.OutlineMenu
Call agentLog.LogAction("After read parameters")
```

We locate the corresponding outline options menu for the selected functional page.

```
'Get the corresponding Outline Menu
NumElements = Ubound(FuncPage)
Call agentLog.LogAction("Elements:= " + Cstr(NumElements))
For i = 0 To NumElements
    Call agentLog.LogAction("into the loop")
    If ( Cstr(FunctionalPage) = Cstr(FuncPage(i)) )Then
        Menu = OutlineMenu(i)
        Set outline = db.GetOutline(Cstr(Menu))
        Call agentLog.LogAction("Outline: " + Cstr(Menu))
        Exit For
    End If
Next
```

After that, we look for the required entry on the outline menu.

```
'Get the required entry
Set entry = outline.GetFirst()

If Not(entry Is Nothing) Then

    Do


        If ( Cstr(entry.Label) = Cstr(Options) ) Then

            'Set the label
            If ( Cstr(TitleChoose)  <> "") Then
                entry.Label =  Cstr(TitleChoose)
                Call agentLog.LogAction("Entry label changed")
            Else
                Call agentLog.LogAction("Entry label not changed")
            End If

            'Set the new URL
            If  (Cstr(URLChoose) <> "") Then

                Call agentLog.LogAction("To modify the URL")

                If ( entry.SetURL(Cstr(URLChoose)) ) Then
                    Call agentLog.LogAction("URL modified")
                Else
                    Call agentLog.LogAction("URL not assigned")
                End If

            Else
                Call agentLog.LogAction("URL not modified")
```

```
                End If

                ' Get the functional choose doc
                Set view = db.GetView("ViewFuncChoose")
                Set doc2 = view.GetFirstDocument

                Call agentLog.LogAction("looking for  the functional choose doc")
                If Not(doc2 Is Nothing) Then

                    Do
                        If (  (Cstr(doc2.FunctionalPage(0)) = Cstr(FunctionalPage)) And
(Cstr(doc2.ChooseName(0)) = Cstr(Options)) )Then

                            'Set the label
                            If ( Cstr(TitleChoose)  <> "") Then
                                doc2.ChooseName =  Cstr(TitleChoose)
                            End If

                            Call doc2.save(True, True)

                            Call agentLog.LogAction("Fuctional choose modified")
                            Exit Do

                        End If
                        Set doc2 = view.GetNextDocument(doc2)

                    Loop While ( Not(doc2 Is Nothing) )

                End If

                Exit Do

            End If

            Set entry = outline.GetNext(entry)

        Loop While Not(entry Is Nothing)

    End If

    Call outline.save()

    Call agentLog.LogAction("End UpdateOutlineEntryWeb")
    Call agentLog.Close

End Sub
```

Once we have located the entry, we check if the entry label and the URL need to be changed. If so, we make the corresponding changes. Using the same criteria, we locate and modify its corresponding “functionalchoices” document.

7.3 Feedback

To encourage users to consult the Web site and be satisfied with it, we want to get their feedback, with questions like these: ”Is the navigation easy? Is

finding knowledge intuitive? Are the response times OK?" Being aware of users' feelings enables us to improve the design of the site.

The portal database Feedback Return Web page has a form containing one Rich Text input field, into which users can type their comments or suggestions about the site. By clicking on the Send button, a memo is sent to the webmaster. The memo body is filled in with this input field. Users can reset their input by pressing the reset button. For a sample, see Figure 76.

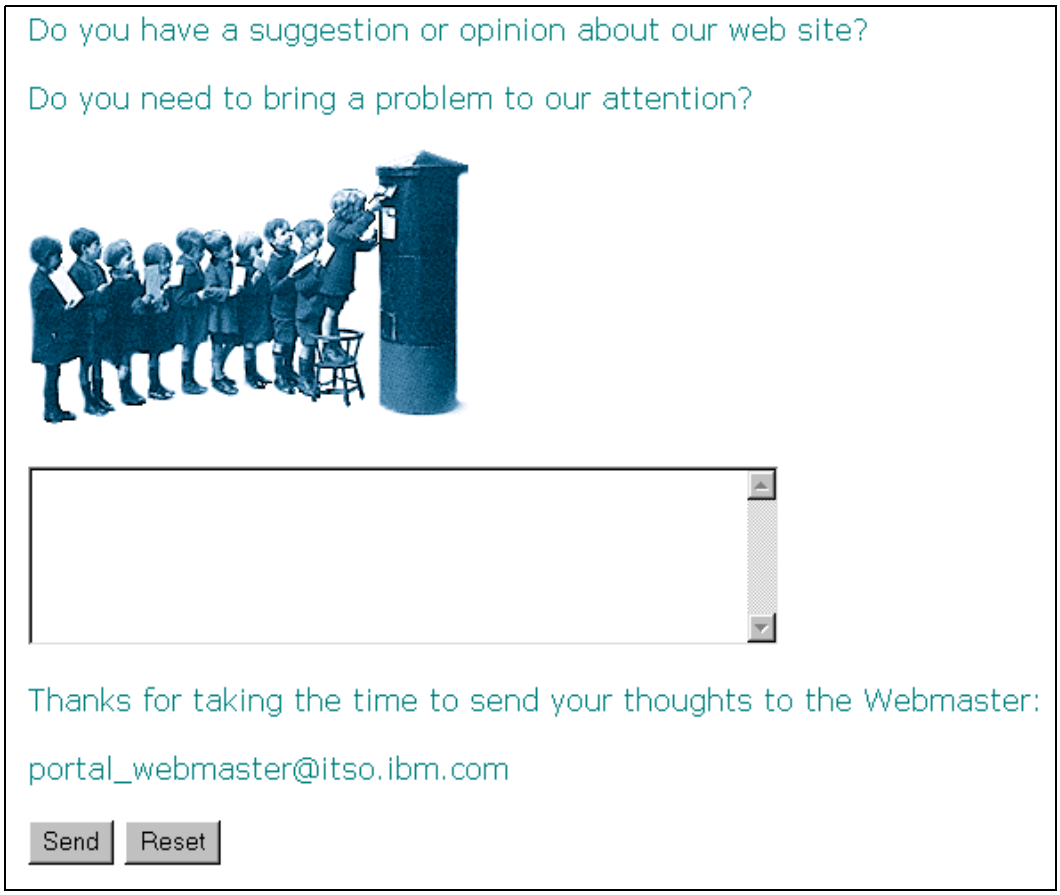

Do you have a suggestion or opinion about our web site?
Do you need to bring a problem to our attention?
Thanks for taking the time to send your thoughts to the Webmaster:
portal_webmaster@itso.ibm.com
Send
Reset

Figure 76. Feedback panel

7.3.1 Designing for feedback

The Feedback Return Web page has been implemented in our portal database by adding the following design elements.

1. Two icons were added:

   - icon_feedback.gif
   - feedback_kids.gif

   From the designer, select **Resources -> Images**, and click the **New Image Resource** button. Enter the path for the image you want to add to the portal design. This is shown in Figure 77.

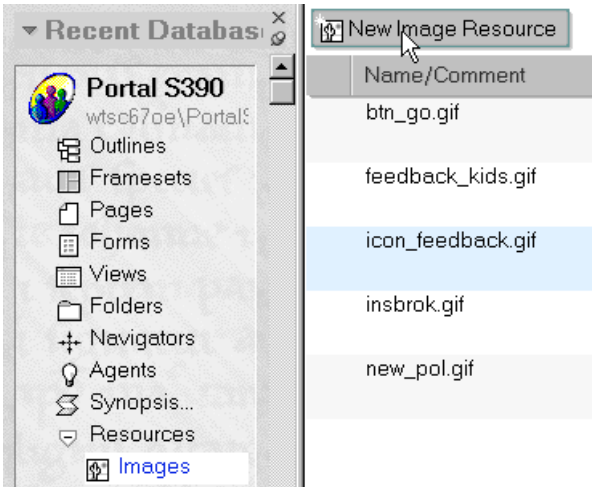


Figure 77. Adding a new image resource

2. The **Feedback** Outline Entry, added to the **PortalMainMenu** outline, is shown in Figure 78 on page 154.

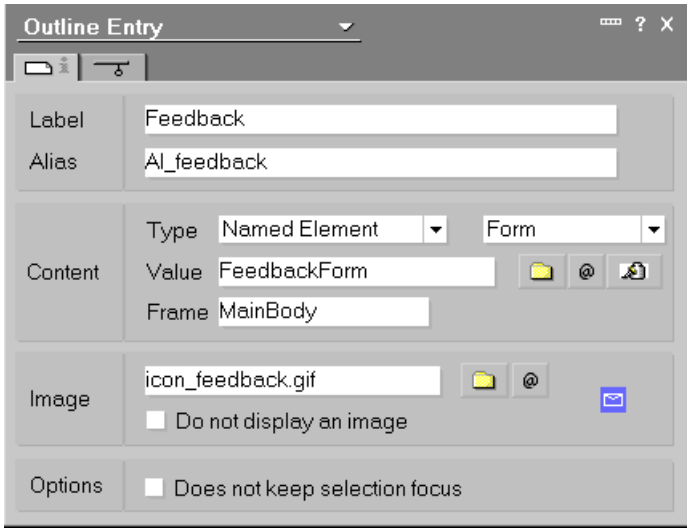


Figure 78. Feedback outline entry

3. One form was added, called FeedbackForm, containing the design elements listed below:

   a. Text representing the message addressed to the Web site users.
   b. One Body text field, hidden from Web browser users.

      This field is the Body of the mail sent to the webmaster by the notes users when submitting their feedback. See Figures 79 and 80.

Figure 79. Body text field

Figure 80. Hiding a field

c. One Rich Text field was added, called RichTextBody; this was hidden from the Notes clients users.

   This field is the Body of the mail sent to the webmaster by the Web browser users when submitting their feedback. See Figure 81.

Figure 81. RichTextBody field

d. One Send button: launches either NotesSendMail or WebSendMail, depending on the client type.
e. One Reset button: clears the input text fields.
f. One SendTo hidden field, containing the webmaster e-mail address where the feedback will be sent.

4. Two agents:

   a. NotesSendMail: This LotusScript agent is designed to get the user message from the Notes form and send it to the Webmaster. See Figure 82 on page 156.

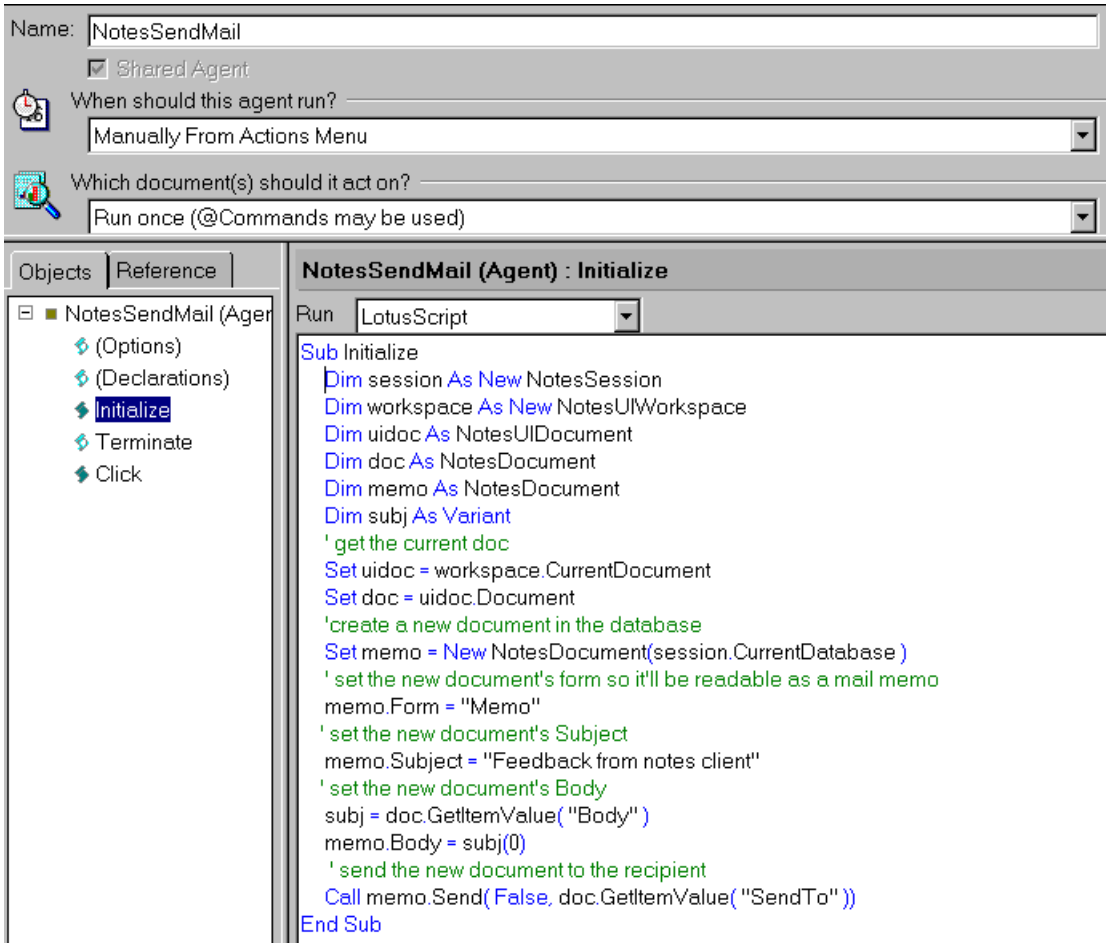


Figure 82. NotesSendMail

b. WebSendMail: This Java agent is designed to get the user message from the Web page and send it to the webmaster. See Figures 83 and 84.

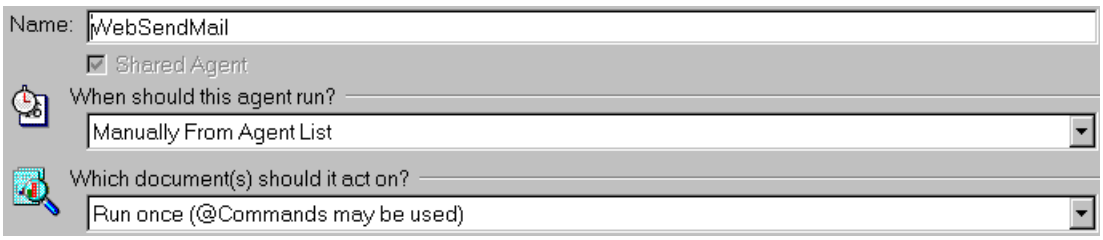

Name: WebSendMail
Shared Agent
When should this agent run?
Manually From Agent List
Which document(s) should it act on?
Run once (@Commands may be used)

Figure 83. WebSendMail

WebSendMail (Agent) : Action

Run: Java

- JavaAgent.java

```
import lotus.domino.*;
import java.util.Vector;
public class JavaAgent extends AgentBase {
 public void NotesMain() {
  try {
      Session session = getSession();
      AgentContext agentContext = session.getAgentContext();
      Database db = agentContext.getCurrentDatabase();
      Document doc = agentContext.getDocumentContext();
      //compose the memo
      Document memo = db.createDocument();
      memo.appendItemValue("Form", "Memo");
      memo.appendItemValue("Subject", "Feedback from Web client");
      memo.appendItemValue("Body", doc.getItemValueString("RichTextBody"));
      Vector v = new Vector();
      v.addElement(session.getUserName());
      v.addElement(doc.getItemValueString("SendTo"));
      memo.setEncryptOnSend(true);
      memo.setSaveMessageOnSend(false);
      memo.setSignOnSend(true);
      memo.sign();
      // send the memo
      memo.send(true, v);
   } catch(Exception e) {
   e.printStackTrace();
    }
  }
}
```

Figure 84. WebSendMail agent

Chapter 8. Integration using Lotus connector products

This chapter covers methods to integrate Domino data and applications with S/390 subsystems using connector products, including DECS and LEI. These provide a declarative interface and do not need user programming. Programmatic connectors are covered in Chapter 9, “Integration using programmatic connectors” on page 175.

For more details on Lotus connector products, see *Lotus Domino for S/390 Release 5: Enterprise Integration Using Lotus Connector Products*, SG24-5682.

8.1 Overview of Domino and enterprise integration

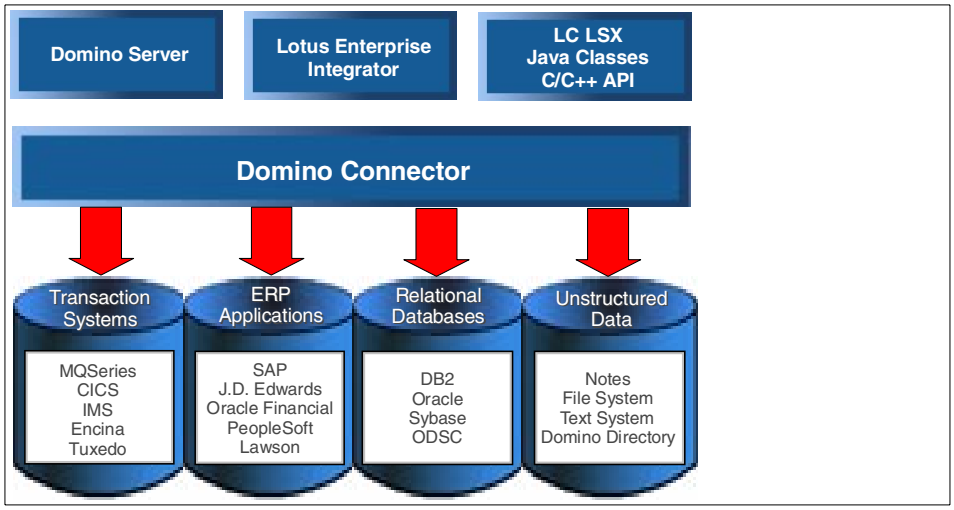


Figure 85. Domino enterprise integration solutions

Lotus provides services for connecting Domino R5 to relational database management systems, transaction processing systems, enterprise resource planning applications, and unstructured data. These services present the user with a common interface to the many enterprise systems. As shown in Figure 85, the basic services are listed below.

Note: This book applies to all platforms, but not all services are available on all platforms. Since we use the S/390 in our examples, we will note which services are not available on the S/390.

© Copyright IBM Corp. 2001

- Domino Enterprise Connection Services (DECS) provides a real-time forms-based interface to enterprise data. DECS is an add-in task bundled with the Domino server. DECS has been available since Domino R4.6.3.
- Lotus Enterprise Integrator (LEI) provides scheduled and event-driven high-speed data transfer and synchronization capabilities between Domino and enterprise systems. LEI ships as a separate product.
- Lotus Connector LotusScript Extension (LCLSX) provides LotusScript access to enterprise systems. The LCLSX is bundled with Domino R4.6.3 and above and LEI 3.0.
- Lotus Connector Java classes provide Java access to enterprise systems using the Lotus connectors. The LC Java classes are available on the Lotus Enterprise Integration Web site and are bundled with LEI 3.0. At this writing, they were not available on S/390.
- Lotus Connector API provides C/C++ access to enterprise systems. The LC API is available on the Lotus Enterprise Integration Web site:

  `http://www.lotus.com/home.nsf/welcome/eizone`

Underlying the above services are individual connectors for the supported enterprise systems.

8.1.1 Lotus connectors

Lotus connectors (LC) exist to permit access to external data sources from Domino for relational database management systems, enterprise resource planning systems, transaction processing systems, directory services, and other services. An overview of the connectors follows.

- Relational Database Management Systems (RDBMS). The following relational database management systems connectors are bundled with DECS and LEI (only DB2 is available on S/390 at this time):
  - DB2
  - ODBC
  - Oracle
  - Sybase
  - EDA/SQL
- Enterprise Resource Planning applications (ERP). The following list identifies the enterprise resource planning connectors. These are all separate products (not available on S/390).
  - J.D. Edwards One World
  - Oracle Financial Applications
  - PeopleSoft

  - SAP R/3
  - Lawson Enterprise/400

- Transaction Processing systems. The following list identifies the online transaction processing (OLTP) system connectors. These connectors are all separate products.
  - CICS (available on S/390)
  - IMS
  - MQSeries (available on S/390)
  - Transarc Encina TXSeries
  - BEA Tuxedo
- Directory services. The directory services connectors in the following list are bundled with DECS and LEI.
  - Domino Directory (available on S/390)
  - Lightweight Directory Access Protocol (LDAP)
  - Novell Directory Services (NDS)
- Other services. The additional connectors in the following list are bundled with DECS and LEI.
  - File System
  - Notes (available on S/390)
  - Text (available on S/390)

8.1.2 Tools for integrating enterprise data and applications

Domino applications can integrate enterprise data or applications using either declarative tools or programmatic tools. The correct tool depends strongly on the solution that is going to be implemented.

Declarative tools should be the first choice, since they enable enterprise data to be integrated without any additional programming. An existing Domino application can be connected to an enterprise database (for example, DB2) after configuring some setup documents.

If you consider using programmatic connector access, additional programming is necessary in order to connect to an enterprise database. At the time of this writing, direct access from Domino for S/390 to enterprise applications such as CICS is only possible using the MQLSX programmatic connectors.

Access to enterprise data can be divided into two different methods. First, real-time access, for which DECS is used. Second, data transfer or synchronization, which is done using LEI. With the programmatic connectors, either method can be implemented.

It is essential to fully understand the way these tools work, in order to integrate them into an enterprise application. All models supply read and write access to the enterprise system. In order to establish a connection to an enterprise database, there has to be an existing Domino application that connects to the enterprise system.

The different models for enterprise database or application access are described in the next sections, including examples that are included in the portal application. For details on this subject, refer to *Lotus Domino for S/390 Release 5: Enterprise Integration Using Lotus Connector Products*, SG24-5682.

8.2 Integrating DB2 data using DECS

DECS comes with the Domino R5 server on S/390 by default. In order to have DECS working properly, some setup is needed during server installation. On Domino for S/390, DECS can connect only to DB2 databases.

8.2.1 Activating DECS during installation

While installing Domino, you must choose the enterprise Domino server in order to install the files needed for DECS. During the configuration of the Domino server, in the Server Audience screen, place a checkmark beside DECS, as shown in Figure 86 on page 163.

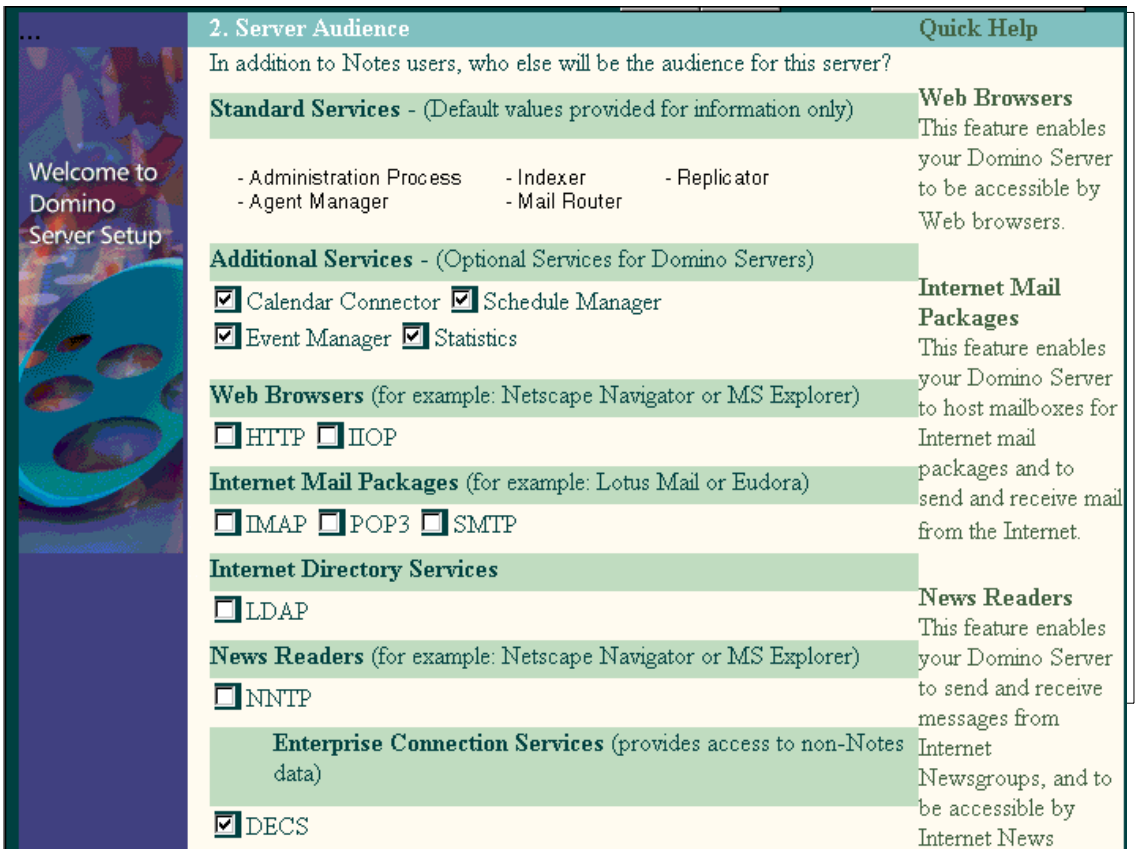

Welcome to Domino Server Setup
2. Server Audience
In addition to Notes users, who else will be the audience for this server?
Standard Services - (Default values provided for information only)
- Administration Process
- Agent Manager
- Indexer
- Mail Router
- Replicator
Additional Services - (Optional Services for Domino Servers)
Calendar Connector
Schedule Manager
Event Manager
Statistics
Web Browsers (for example: Netscape Navigator or MS Explorer)
HTTP
IIOP
Internet Mail Packages (for example: Lotus Mail or Eudora)
IMAP
POP3
SMTP
Internet Directory Services
LDAP
News Readers (for example: Netscape Navigator or MS Explorer)
NNTP
Enterprise Connection Services (provides access to non-Notes data)
DECS
Quick Help
Web Browsers
This feature enables your Domino Server to be accessible by Web browsers.
Internet Mail Packages
This feature enables your Domino Server to host mailboxes for Internet mail packages and to send and receive mail from the Internet.
News Readers
This feature enables your Domino Server to send and receive messages from Internet Newsgroups, and to be accessible by Internet News

Figure 86. Selecting DECS during configuration of the Domino server

For detailed information about installing and configuring Domino in a S/390, environment, refer to *Lotus Domino for S/390 Release 5: Installation, Customization and Administration*, SG24-2083.

Domino Enterprise Connection Services (DECS) starts running as you bring up the Domino task, once you complete the setup process. The Domino console lists the tasks, showing the status of the DECS server task.

8.2.2 Introduction to the concepts of DECS

DECS is a data access tool that can be used for connecting Notes databases to DB2 enterprise databases.

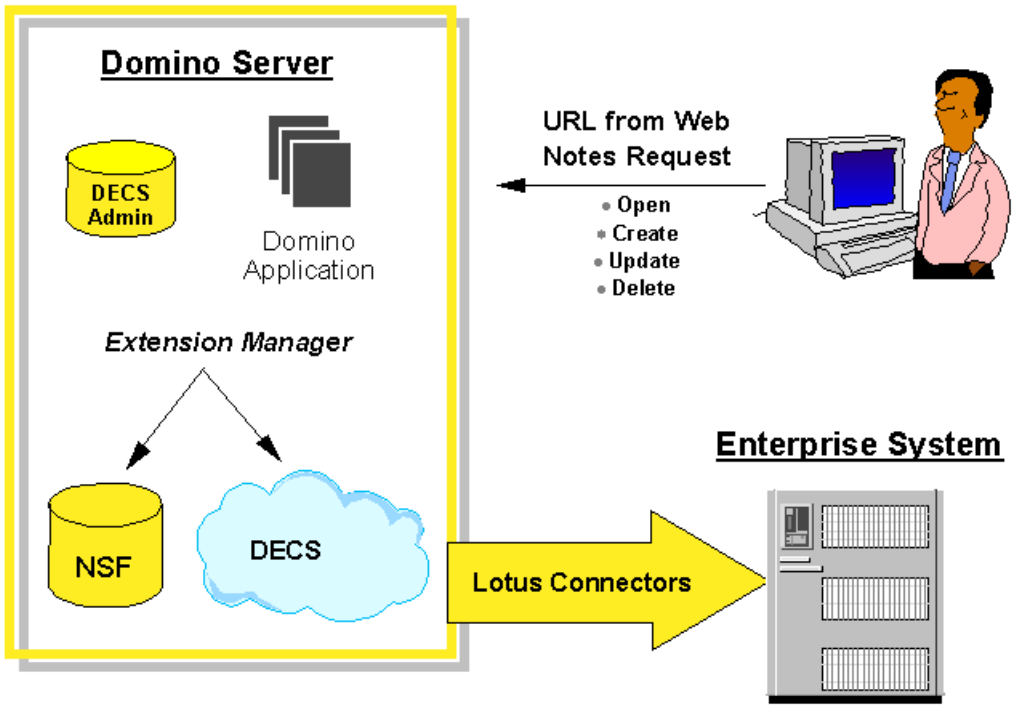


Figure 87. How DECS works

The DECS Model: Declarative real-time access to enterprise data using a fixed interface.

DECS connects to enterprise databases such as DB2. For each row in the database, at least one key field must coexist in the connected Notes database at all times, so partly redundant data is necessary. The rest of the fields of a document are retrieved or updated synchronously from the enterprise system. DECS stores an internal relationship table between each row in the enterprise database table and each Notes document in the Notes database.

The event-driven activation of the connection to the enterprise data for reading, writing or deletion is done by the Domino extension manager module using events monitoring. The open, update, create, and delete events on Notes documents are available as illustrated in the figure above. The relationship must be initialized for each single document or once for all documents in order to work. The connected enterprise tables are statically defined for each connection. DECS connects to one table per connection.

8.2.3 The sample portal customer application

In our portal, we included a customer database. The enterprise database is a DB2 table and the front end to the DB2 table is a Notes database. The records of the database can be filled, altered, and deleted using the Notes database, which manipulates the DB2 table using the DECS tool.

The application can be accessed through a Web browser or Notes client.

Note

We experienced an error using DECS with the Web browser. Retrieval of the data did not work at all times. The error is reported and should be fixed in succeeding releases.

8.2.3.1 The DB2 part of the application

We used the following SQL to create the DB2 table shown in Figure 89 on page 166:

```
CREATE TABLE DSN8610.CUSTOMERS (
      CUSTNO   CHAR(6) NOT NULL,
      COMPNAME CHAR(20),
      FIRSTNME CHAR(20),
      LASTNAME CHAR(20),
      GENDER   CHAR(1),
      HOBBIES  CHAR(40),
      DEPT     CHAR(40),
      PHONE    CHAR(20),
      MAIL     CHAR(40),
      INFOS    CHAR(200) )
         IN DSN8D61A.DSN8S61R;
```

Figure 88. SQL to create a DB2 table

DB2G is the name of our DB2 subsystem on the S/390, DSN8610 is the default identifier which set up the table, DSN8D61A is the tablespace, and DSN8S61R is the name of the database.

The Custno Column will be used as the DECS key, so a null entry should not be allowed.

8.2.3.2 The Domino part of the application

We created a simple application to show how DECS works. The customer form was created for the both the Notes client and Web access. The form

functionality is identical for Notes and Web. It is also used for field mapping in the DECS tool.

The fields are all text fields. Gender and hobbies are multivalue fields.

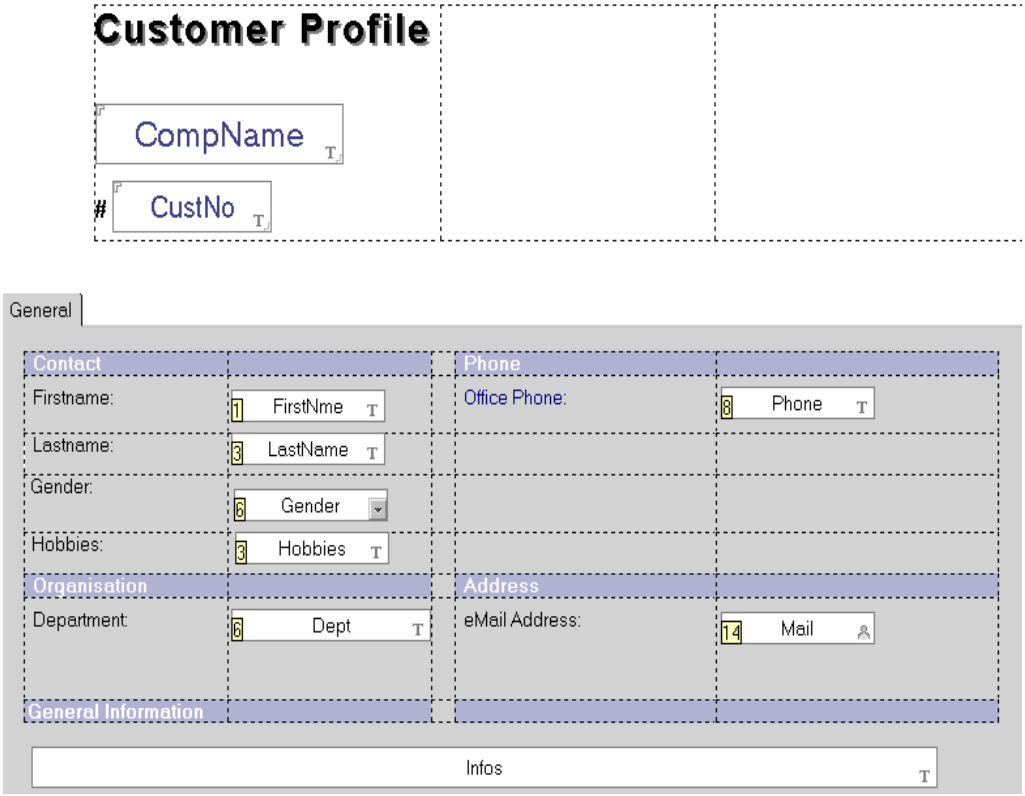


Figure 89. Customer form

For all the views we used, we activated the “use applet in the browser” flag in the view properties, so they look like Notes when using the Web browser. This allows users to use either interface comfortably. Figure 90 on page 167 shows where to set this flag.

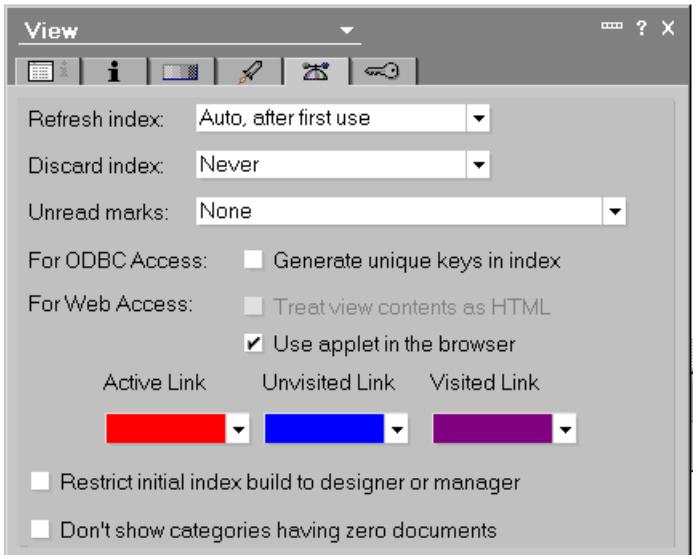


Figure 90. Activate "Use applet in the browser" for views

This application also incorporates a page with an embedded outline for navigation.

Note: If you would like to use this application as a starting point, we have included the source code on the ITSO web site. See Appendix B, "Using the additional material" on page 263.

8.2.3.3 The DECS settings

The DECS administrator database is, by default, installed on your server if you install it as we have described. In this database, we created a connection document, which pointed to the created customers DB2 table. See Figure 91 on page 168.

If DB2 is configured correctly on the S/390 so the Domino server with the DECS task can connect to it, you can fill out the database subsystem name, user name, and password that should be used by the Domino server. If you then press the owner or name button, DECS will browse the DB2 database through the Domino server and list all available tables.

If it doesn't work, the connection or the interface to DB2 are probably not set correctly. For details, refer to *Enterprise Integration Using Lotus Connector Products,* SG24-5682 or the DECS help database.

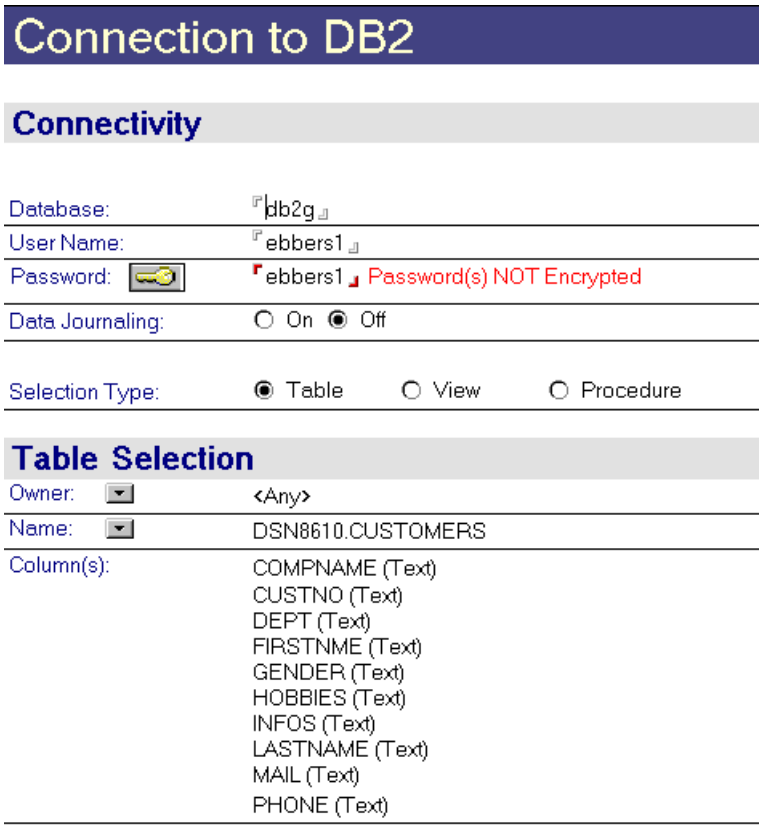


Figure 91. DECS customer connection document

Using this connection, we created an activity document that controls the way DECS does the enterprise database retrieval. The field mapping we used for the Notes form Person of the customers database and the customers DB2 table can be seen in Figure 92 on page 169.

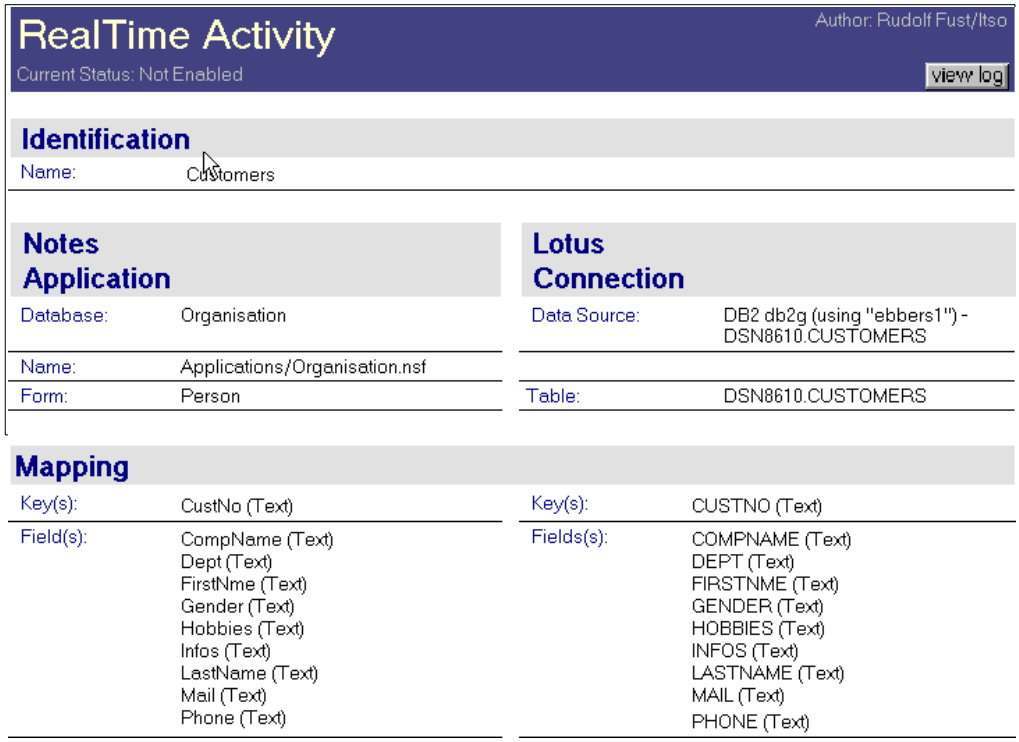


Figure 92. The DECS customer activity document

In the activity document, we chose all available events, as shown in Figure 93 on page 170, so the create, update, open, and delete event will access the enterprise DB2 database through the extension manager and DECS.

In the general options, we defined the fields Dept and Lastname to be saved in the Notes documents, in addition to the key, which is stored by default. This gave us enough information in a view to browse the document that retrieves the rest of the content from the DB2 table.

The settings are shown in the following partial view of the Customer Activity document.

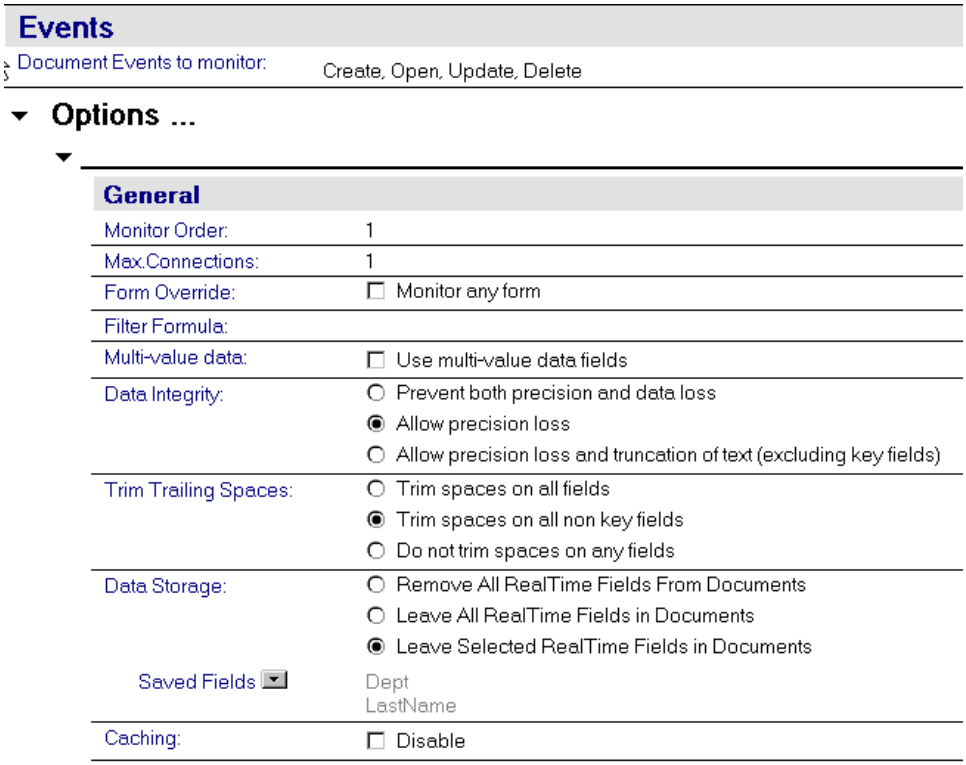


Figure 93. Events and options in the DECS activity document

If the DB2 database is empty, you start the activity with the start button in the DECS administrator navigator. The DB2 table is then filled by the user, using a Notes client, for example. When a document is composed, filled with content, and saved, the extension manager acts on the created event and puts the additional key data into the DB2 table.

The other way to start out is with a filled DB2 table. An initialization of the Notes database with the keyfields is executed by the Initialize Fields action as shown in Figure 94 on page 171.

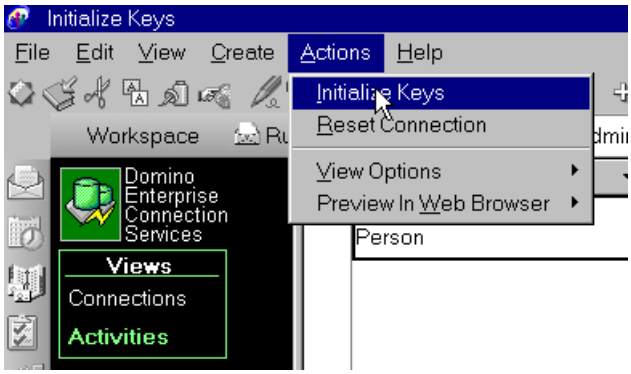


Figure 94. DECS key initialization in the Notes database

The key fields must always exist on both sides, in the Notes database and in the DB2 table. The Notes database must contain at least one field, the key field for each document. As soon as the document is opened, the rest of the fields are loaded from the DB2 table through the open event of the extension manager and the DECS task. The user does not notice that the fields were loaded from a DB2 table. It happens quickly and is like working with a normal Notes application, such as in Figure 95.

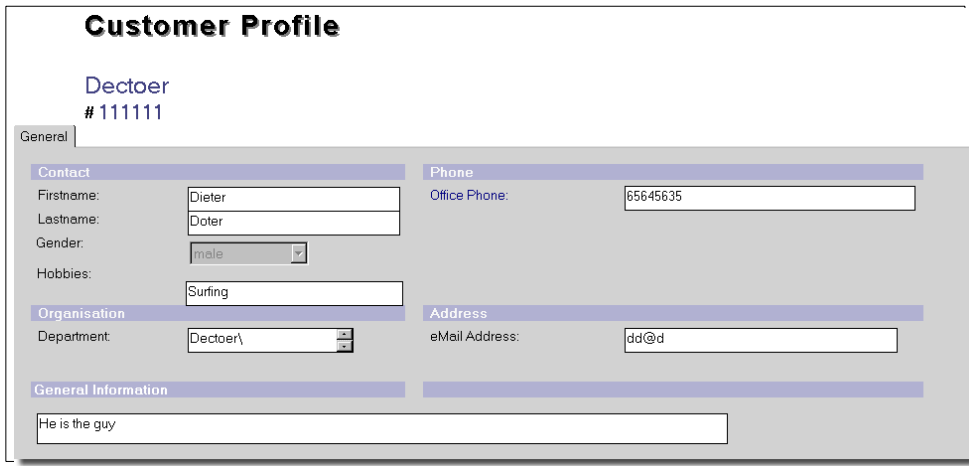


Figure 95. Opening a document using DECS

You can tell the difference only if the DECS activity is not started. When the document is saved again after update, the fields (except the key) are updated in the DB2 table using the update event of the extension manager and deleted from the Notes document.

This mechanism also applies to Figure 89.

8.3 Integrating DB2 data using LEI

Lotus Enterprise Integrator (LEI) is a utility that does not come with the Domino server; it has to be purchased separately. LEI can set up a connection between any standard Notes database and an enterprise system, such as a relational table. On S/390, LEI can set up connections only between Notes databases and DB2 databases.

8.3.1 The concepts of LEI

LEI data transfer is initiated manually or in a time-triggered manner. It usually works as a triggered batch job. LEI also contains a real-time component like DECS. However, users are strongly encouraged to use the DECS tool for real-time activities.

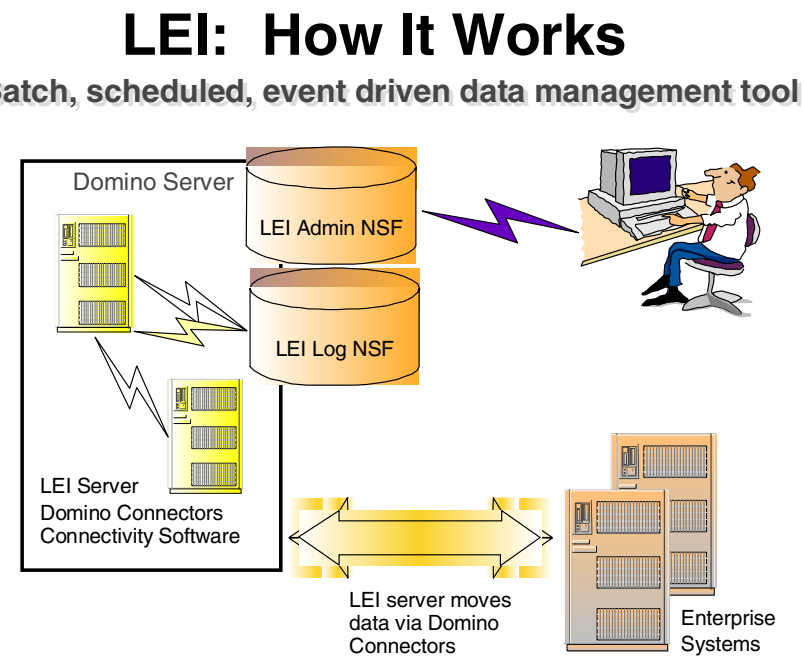


Figure 96. How LEI works

The LEI Model: Declarative batch job-based data transfer tool.

LEI is a asynchronous data transfer tool. It can transfer large volumes of data in either direction between any two data sources, such as Domino databases and DB2 tables. The transfer can be used for redundant data storage in two data sources. It can also be used for moving data from one data source to another. The data transfer generally happens asynchronously in the background in a scheduled or manually-activated execution. The connected tables are statically defined for each connection, usually one table per connection.

The mapping between the Notes database and the DB2 table is done like in DECS. The relation is set up between the fields of a form of a Notes database and the columns of a DB2 table.

In this redbook, we did not develop an application using LEI. If you want to see an example of how to use LEI for enterprise integration or learn about it in more detail, refer to *Lotus Domino for S/390 Release 5: Enterprise Integration Using Lotus Connector Products*, SG24-5682.

The reason we left out LEI is that from the application point of view there is no difference if it is a pure Domino application where the documents are created in Domino or if the documents came from an enterprise database. In both cases, the database design elements would be the same. If you are using the database in a portal, it is just like using a standard Notes database.

LEI only puts data into an existing database. It can insert the documents as a direct data transfer into the database or as a replication-like synchronization between the enterprise data and the Notes documents. Each Notes document corresponds to a record in a table of a relational database.

The only thing you have to think about is which server takes care of the document transfers and the connection to the enterprise database. With this information, you can develop a distribution concept for the Notes database. One advantage of LEI is that the Notes documents can be distributed like a normal Notes database. No online connectivity to a Domino server or an enterprise system is needed.

Chapter 9. Integration using programmatic connectors

In this chapter, we describe how to connect Domino data and applications to enterprise S/390 subsystems by using programmatic connector utilities. The programmatic method can offer advantages in flexibility and performance, but it requires you to write application code.

For more details on Lotus connector products (standard and programmatic) including sample code, see *Lotus Domino for S/390 Release 5: Enterprise Integration Using Domino Connector Products*, SG24-5682.

9.1 Integrating DB2 data using programmatic connectors

Programmatic connectors for Domino on S/390 can be divided into LCLSX, LS:DO, and C-API connectors. Since Lotus focuses on the LCLSX technology for strategic plans, we also focus on it in this redbook.

The LCLSX connector code comes with the standard Domino Server or client code. It forms the basis for both DECS and LEI. It is usable instantly after installation just by including a statement in the LotusScript codes that use it. For Domino on S/390, the LCLSX connectors can connect only to DB2 databases.

While DECS and LEI offer declarative access to enterprise databases, the use of the LCLSX connectors requires coding.

You can also program your own connectors using a LotusScript Extension (LSX). There is a Lotus Connector API available from Lotus that permits development of additional Lotus Connectors for Domino.

9.1.1 The concepts of LCLSX

The programmatic connector model provides a flexible real-time tool for accessing or transferring low volumes of data.

The programmatic connector model provides data access to enterprise databases such as DB2. The model gives very flexible access to the enterprise system. Synchronous real-time access as well as asynchronous scheduled access can be realized in all varieties using all possible Notes events and activation functions. The connectors can also be used from a Notes client; this requires an additional enterprise client application (such as DB2 Connect) on the client workstation. The programmatic connections code can be embedded in LotusScript code. Different enterprise tables can be flexibly handled with the same object model.

© Copyright IBM Corp. 2001

The LotusScript Extension for Lotus Connectors (LCLSX) extends use of the Lotus Connectors to LotusScript. The programming model is independent of individual connectors, eliminating the need to learn each system, but allowing experienced users to access specific connector features. Using this LSX, developers can access external data natively, using the same object model to make calls syntactically to a variety of enterprise systems. This reduces the learning curve for integrating Domino with enterprise data sources.

Through the LCLSX, Notes and Web applications can retrieve and act upon data from agents, during document events, or by clicking a button. The LCLSX can be used alone or in conjunction with Lotus Enterprise Integrator (LEI). Together these technologies provide programmatic and declarative access to external data.

At the time of this writing, Domino R5 on S/390 provides the following standard set of connectors:

- DB2
- Notes

9.1.1.1 LCLSX classes

The LCLSX classes are an extension to the standard Notes classes. They have to be loaded using the `Use` statement in the options section of each code segment. The classes are by default installed on every Notes client or Domino server and ready to use. There are four LCLSX classes:

- LCSession
- LCConnection
- LCFieldlist
- LCField

In addition to the four classes, there are four advanced data types:

- LCStream
- LCNumeric
- LCCurrency
- LCDatetime

Place the following statement in the Options event of your LotusScript program to make the LCLSX classes available:

```
Uselsx "*LSXLC"
```

The Figure 97 on page 177 shows the relationship between the different LCLSX classes.

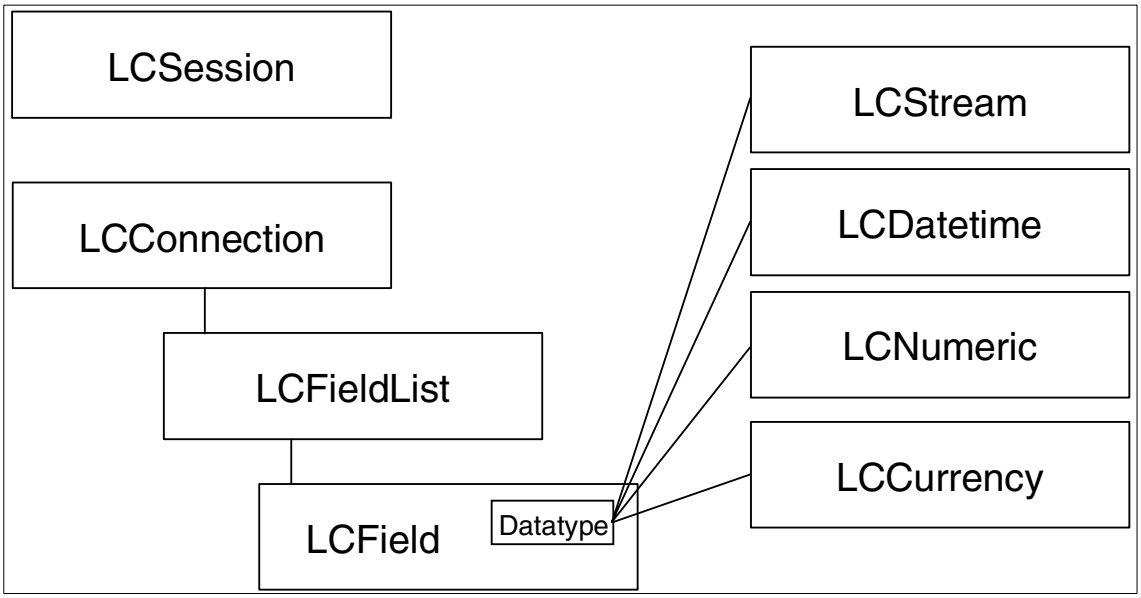


Figure 97. Relationship between different LCLSX classes with data types

9.1.1.2 LCSession

The LCSession class provides error information useful in error handling. It also provides for query and lookup of available connectors. You do not need the LCSession class in order to connect to a data source, but it is recommended because it gives good error feedback.

9.1.1.3 LCConnection

The LCConnection class represents an instance of a Lotus connector. This class provides query and data access to the external system. Multiple instances of an LCConnection class can be allocated to a single connector.

LCConnection properties depend on the specific connection. Typical properties include LCConnection.Database, LCConnection.UserID, and LCConnection.Password.

9.1.1.4 LCFieldlist

The LCFieldlist class is the primary class for manipulating data through a connection. It binds a group of fields together with names and a default order.

When you generate a result set and pass in an empty fieldlist, the connector automatically populates the fieldlist. For each data element of the result set, the fieldlist receives the element's name and an LCField object of the

corresponding data type. The result set can be controlled by manually building the fieldlist before generating the result set, or by using the FieldNames property of the connection.

The Select and Call connection methods use an optional fieldlist of keys or parameters to restrict the result set. You construct this fieldlist manually and pass it to the Select method. You construct a key or parameter list by appending or inserting names and data types to the list. These methods create fields in the fieldlist and return the fields for further manipulation. The fields are then given values and, using field flags, can be given conditions such as greater-than and not-equal.

The typical use of connectors is to gather, create, and modify data in the enterprise system. For example, a Domino application has a number of data fields on a form. Once user input is complete, a button activates a script to update corresponding information in an external data store.

For this redbook, we concentrate on scripts running on a Domino for S/390 server. The script handles all the database activity on the Domino server, so only a thin client without any database connectivity software is needed. It is possible to execute the scripts locally on a Notes Client, but in that case connectivity software such as DB2 Connect is needed on each Notes client that accesses the enterprise system.

9.1.2 The sample portal invoice application

We included an invoice database in the portal. The enterprise database is a DB2 table and the front end to the DB2 table is a Notes database. The records of the database can be fetched online, inserted, updated, and deleted without any redundantly stored information in the Notes database. In normal operation the database is completely empty. It is filled only with retrieved DB2 records, which are loaded into Notes documents. The documents can be worked with like any other Notes documents. The only difference is that they are not updated or inserted in the Notes database, but directly in the DB2 table and automatically deleted after the DB2 update in Notes.

The application can be used with the Notes client or the Web browser. Web browser access is, in this case, only for fetching and viewing data. Notes client access has the full fetch, update/insert, and delete functionality.

9.1.2.1 The DB2 part of the application

We used the following SQL to create the DB2 table:

```
CREATE TABLE DSN8610.INVOICES (
       INVNO     CHAR(6) NOT NULL,
       COMPNAME  CHAR(20),
       CUSTNO    CHAR(20),
       DATE      CHAR(10),
       ITEMS     CHAR(200),
       SUM       DECIMAL(8,2),
       PAID      DECIMAL(8,2),
       STATUS    CHAR(20),
       COMMENT   CHAR(200) )
          IN DSN8D61A.DSN8S61R;
```

DB2G is the name of our DB2 subsystem on the S/390, DSN8610 is the default identifier which set up the table, DSN8D61A is the tablespace, and DSN8S61R is the name of the database.

9.1.2.2 The Notes part of the invoices portal application

The Notes part of the application is a Notes database front end to the enterprise DB2 database. Both databases can be fully managed with the Notes application.

The Notes application consists basically of the invoice form, a page with an embedded outline, views and some agents using LCLSX for the enterprise data management. The DB2 database either can be filled by the Notes application using the update/insert agent, or it can already exist filled with data. The Notes database is empty in initial status.

When a user wants to retrieve an invoice, he clicks **Search invoice** and fills in the invoice number. All invoices with that number are retrieved and will be visible in the Notes view as standard Notes documents, as shown in Figure 98.

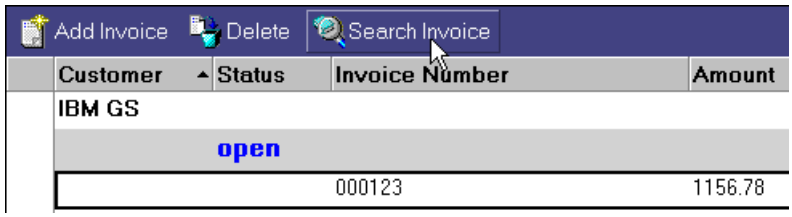


Figure 98. Searching for an invoice in the LCLSX application

The document can now be viewed or edited. After changing the document, it can be updated in DB2 or deleted, using the action buttons shown in Figure 99.

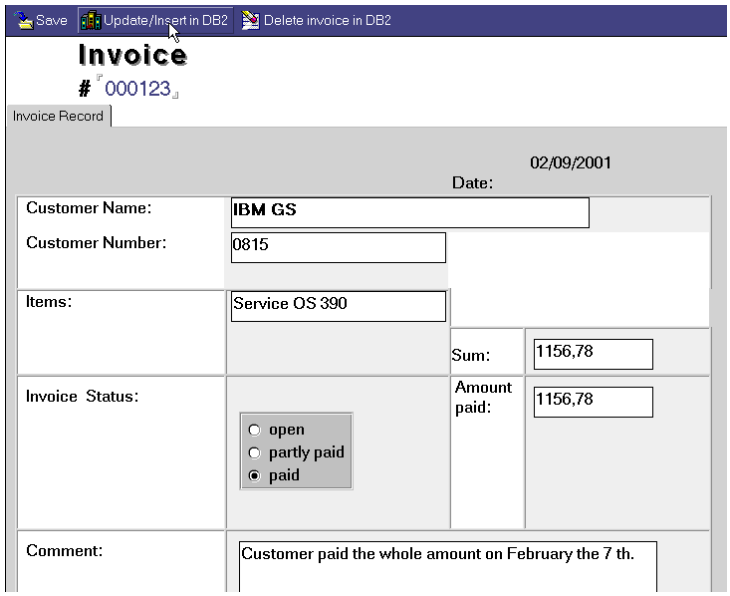


Figure 99. Updating the invoice using LCLSX technology

After each operation with DB2, the document is deleted from the Notes database.

Each of the button-triggered actions calls an agent on the database. These reside on the Domino server and use the agent.runonserver method. The parameter is handed to the agent using the *agent parameter* document property.

9.1.2.3 Example of an LCLSX agent

To show how programming using LCLSX is done, we discuss a sample agent from the invoice application. This agent retrieves invoices by number and puts them as Notes documents in the database. This version of the agent is executed from a Notes client. You can develop the same functionality for the Web with some slight differences. For details, refer to *Enterprise Integration Using Lotus Connector Products*, SG24-5682.

The activation button in the Notes client

The agent is activated by a button with the following script code. This is the major difference from the Web agent solution.

```
Sub Click(Source As Button)
   Dim session As New NotesSession
   Dim db As NotesDatabase
   Dim Agent As NotesAgent
   Dim ParmDoc As Notesdocument

   Set db = session.CurrentDatabase
   key$ = Inputbox$("Enter the Invoice Number")
```

In the following block, the parameter document is created, the invoice number is written in the LCkey field, and the document is saved to the current database residing on the server. The agent must run on the same server. The agent reads the LCkey field after the document is saved, to get the key for its retrieval.

```
   ' Create Parameter Document for Agent
   Set ParmDoc=DB.Createdocument
   ParmDoc.Subject = "Agentparameterdoc"
   ParmDoc.LCkey = key$
   Call ParmDoc.Save(True, False)
```

In the following lines, we call the Agent that executes the fetch. We include the ID of the parameter document containing the invoice number as a parameter in the call.

```
   ' Get Agent and Run it on the Server
   Set Agent = db.GetAgent( "Get Invoices")
   Print "     . . .  DB2 Fetch Agent running "
   status = Agent.RunOnServer(ParmDoc.NoteID)
   If status = 0 Then Print "  DB2 Data Fetch Agent executed successfully"

End Sub
```

The activation technology used in the Web browser

For Web access, we developed a small form in the invoice database. It can be included in a frame of the sales function portal page, for example. This form can be used from Web and Notes clients. It is shown in Figure 100 on page 182.

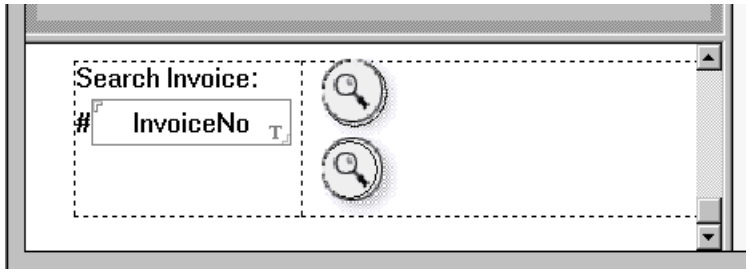


Figure 100. Small frame for invoice search from Web and Notes

The upper hotspot button is for Web access and is hidden in Notes. The lower button has the functionality we described for the Notes client and is hidden on the Web.

The Agent to search for invoices from the Web is activated using the following @commands.

```
@Command([FileSave]);
@Command([ToolsRunMacro];"Get Invoices Web");
@Command([FileCloseWindow]);
@Command([OpenView] ; "Invoices by Customer"  )
```

The code causes the document to be saved, so the agent can read the invoice number parameter using the document context. The Get Invoices Web agent is started and the Invoices by Customer view is launched to view the search results. The parameter documents would have to be deleted, for example using a scheduled agent.

In the example of the Notes invoice search agent, we now discuss the code where the LCLSX connectors are being used. The Get Invoices Web agent code is included in Appendix A, “Code for the portal applications” on page 237. The major difference is the handling of the parameter document. The rest of the code is similar. For easier understanding, we completely separated the code into two agents.

The agent for invoice search from a Notes client

In the options of the agent, you must include this line to load the LCLSX connector codes:

```
Uselsx "*LSXLC"
```

The agent can be scheduled in whatever method you choose. Check the server document security for executing LotusScript agents for the users, even though the agent is executed on the server. The error handler is initialized

and the following Notes Objects, including the agent parameter document, are declared:

```
Sub Initialize
   On Error Goto Handler
   ' Declare Notes Objects
   Dim NSession As New NotesSession
   Dim DB As NotesDatabase
   Dim Agent As NotesAgent
   Dim count As Long
   Dim SQLString As String
   Dim hr_id As String
   Dim EmpDoc As NotesDocument'Fetched Row from DB2 Table
   Dim ParmDoc As NotesDocument       'Parameter Doc from Agent created by
Button
```

The following LC Objects are declared for containing the key information and the resultset information. The “FLD...” fields refer to the DB2 columns.

```
   'Declare LC Objects
   Dim Session As New LCSession
   Dim src As LCConnection
   Dim FieldOut As New LCFieldList
   Dim KeyList As New LCFieldList
   Dim FLDInvNo As LCField
   Dim FLDCompName As LCField
   Dim FLDCustNo As LCField
   Dim FLDDate As LCField
   Dim FLDItems As LCField
   Dim FLDSum As LCField
   Dim FLDPaid As LCField
   Dim FLDStatus As LCField
   Dim FLDComment As LCField
   Dim kfld As LCField
```

The following lines initialize the parameter document from the agent and read the key parameter for the select.

```
   'Get Notes Docs
   Set DB = NSession.currentdatabase
   Set Agent = NSession.currentagent

   'Read key Parameter from Agent Parameter Document
   Set ParmDoc = db.GetDocumentbyID(agent.ParameterDocID)
   Print "Parm Document opened, NoteID = "+ ParmDoc.NoteID
   tkey = Parmdoc.GetItemValue("LCKEY" )
   Print "key " + tkey(0) + "fetched"
```

The connection to the DB2 table is established in the following lines. After the connection is created, all of its properties are accessible to customize the connection to the target system. Common properties include database and server, user ID, and password. Properties are not case-sensitive.

```
'Establish DB2 Connection
Session.ClearStatus

Set src = New LCConnection ("db2") ''Create Connection object

src.Database = "DB2G"
src.Metadata = "DSN8610.INVOICES"
src.UserID ="EBBERS1"
src.Password = "EBBERS1"
Print "Establishing Connection"

src.Connect ''Connect to db2 data source 'DB2G'
Print "Connection established"
```

This selection is accomplished by creating a key list. The default key flag, LCFIELDF_KEY, indicates an exact match. If an inequality, such as 'greater-than' or 'like' is needed, then the field's flags property would be ORed with the corresponding constant. (In all cases, key fields must have the LCFIELDF_KEY constant in addition to any optional conditional flag constants.) The key fieldlist which defines the column and the value used to retrieve the rows is created in the following lines. The value was read before from the agent parameter document.

```
'Create key fieldlist
Set kfld = keylist.append("INVNO", LCTYPE_TEXT)
kfld.Flags = LCFIELDF_KEY

kfld.value = tkey(0)
Print "Key is "+ tkey(0)
```

The Select connection method creates a result set of all records from the external system which match the keylist. If the LotusScript keyword *Nothing* is substituted for the keylist, then all records of the specified metadata would be selected. In this case, all records from the DB2 table are selected.

Here we use our previously created key fieldlist. We match the LC Fields to the columns of the resultset we will receive.

```
'Do a keyed Select on the table
count = src.Select(keylist,1,fieldout)

'Set LC Fields
```

```
Set FLDInvNo = Fieldout.Lookup ("INVNO")
Set FLDCompname = Fieldout.Lookup ("COMPNAME")
Set FLDCustNo = Fieldout.Lookup ("CUSTNO")
Set FLDDate = Fieldout.Lookup ("DATE")
Set FLDItems = Fieldout.Lookup ("ITEMS")
Set FLDSum = Fieldout.Lookup ("SUM")
Set FLDPaid = Fieldout.Lookup ("PAID")
Set FLDStatus = Fieldout.Lookup ("STATUS")
Set FLDComment = Fieldout.Lookup ("COMMENT")
```

The Fetch connection method reads the data from the external system into the result fieldlist (resultset). The individual data values may be accessed from a fieldlist using the expanded class properties. For each field in a fieldlist, there is a property with the corresponding name. This property is an array of values using the closest available LotusScript data type to match the LCLSX data types.

The following loop goes over the Resultset and issues a Fetch as long as any data is left. For each successful fetch, a document is created in our sample application.

```
'Loop over Resultset
While (src.Fetch (Fieldout) > 0)

   'Create Document for each Row / Resultset entry
   Set empdoc = db.createdocument
   empdoc.form= "Invoice"
   empdoc.InvNo = FldInvno.text(0)
   empdoc.CustomerName = FldCompname.text(0)
   empdoc.Customernumber = FLDCustno.text(0)
   empdoc.Date = FLDDate.text(0)
   empdoc.PItems = FLDItems.text(0)
   empdoc.Sum = FLDSum.text(0)
   empdoc.Paid = FLDPaid.text(0)
   empdoc.Status = FLDStatus.text(0)
   empdoc.Comment = FLDComment.text(0)
   Call empdoc.Save (True, False)
   count = count + 1
Wend
```

Finally, we delete the agent parameter document and disconnect from DB2.

```
ret = ParmDoc.Remove(True)    'remove Agent Parameter Document
src.Disconnect ''Disconnect from db2 data source 'DB2G'
Exit Sub
```

The rest is error logging.

```
Handler:
   ''Error Handler
   If ( session.status <> LCSUCCESS) Then
      Print session.GetstatusText, 0, _
      + "The following Lotus Connector error has occurred."
   Else
      Print Error$, 0, _
      + "The following LotusScript error has occurred."
   End If
End Sub
```

A slightly different agent is used for fetching from the Web.

An agent which runs on the Domino server is always difficult to debug in case of errors. Use the log and print statements. For further tips refer to *Lotus Domino for S/390 Release 5: Enterprise Integration Using Domino Connector Products*, SG24-5682.

The agent code of the update/insert and the delete agent is in Appendix A, “Code for the portal applications” on page 237.

9.2 Connecting to enterprise applications with MQLSX

The MQSeries LotusScript Extension (MQLSX) can be used in any environment supporting LotusScript. It provides a set of classes incorporating the full power of the MQSeries Message Queueing Interface (MQI), giving your LotusScript applications the ability to interact with any MQSeries-enabled application in your enterprise, as illustrated in Figure 101 on page 187. The clients can be Notes-based as well as Web-based. In our environment, the MQLSX code runs as an agent on the Domino for S/390 server.

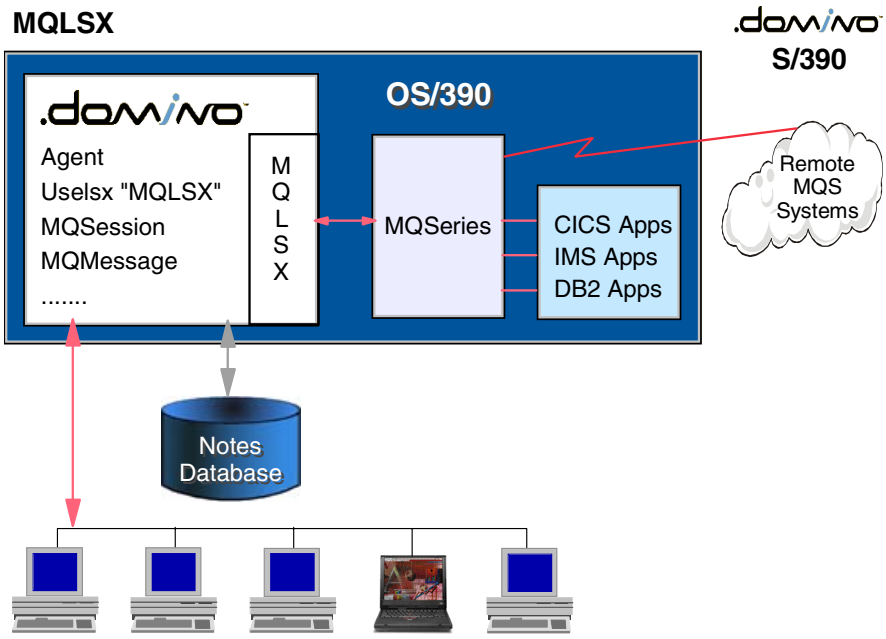


Figure 101. MQLSX model

MQSeries is the IBM message queuing middleware product. It allows two or more applications on like or unlike systems to talk by sending messages. MQSeries uses a combination of programming (MQSeries API or MQI), queues (MQSeries Queue Manager or MQM), and channels (MQSeries Channel Architecture or MCA) to allow this exchange to occur.

For further details on MQSeries, visit this IBM Web site:

`http://www.software.ibm.com/ts/mqseries`

The MQSeries LotusScript Extension allows you to integrate your Domino application with MQSeries, which must be installed on your Domino server. For development purposes it also must be installed on your Notes client.

In this redbook, we will focus on the server installation. The version and type of MQSeries you install is up to you and your installation architecture. As for the type of MQSeries product to install, this depends on your infrastructure and integrity requirements. You can either install the MQSeries thin client or the full MQSeries server with Domino. There are advantages to either decision, and each situation must be explored individually. These considerations are not discussed in this section, as the advantages and

disadvantages are dynamic and depend on the systems you are using and your network infrastructure (the topology, architecture, and protocol).

9.2.1 The MQSeries LotusScript Extension

The MQSeries LotusScript Extension (MQLSX) provides a tight link through MQSeries to transaction-based systems, allowing seamless integration of business processes across Domino and transaction systems.

MQLSX enables you to make MQSeries calls directly from your LotusScript code. Figure 102 shows the architecture of the MQSeries LSX.

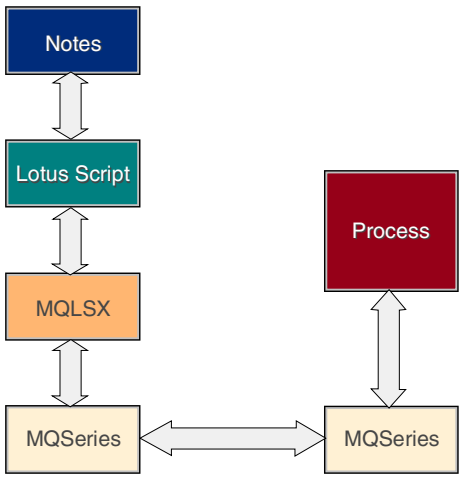


Figure 102. MQLSX architecture placement

With the MQSeries LSX, you can develop applications that integrate the Domino environment with transaction-based applications and legacy system data through MQSeries. This allows you to access enterprise application logic, not just the data, through a variety of platforms, since you have direct integration between Domino and MQSeries.

9.2.1.1 Class overview

There are seven MQSeries LSX classes:

- MQSession
- MQQueueManager
- MQQueue
- MQProcess
- MQMessage
- MQGetMessageOptions
- MQPutMessageOptions

In addition, the MQSeries LSX also provides predefined LotusScript constants, such as MQFMT_NONE, that are used by the classes. The constants are a subset of those defined in the MQSeries C header files, namely CMQC*.H, with some additional IBM MQSeries LSX reason codes.

These classes can be categorized into three groups:

1. The first group contains only the MQSession Class. It provides a root object that contains the status of the last action performed on any of the MQSeries LSX objects.
2. The second group contains the MQQueueManager, MQQueue, and MQProcess classes. They provide access to the underlying IBM MQSeries objects. Methods or properties defined in these classes result in calls being made across the MQI.
3. The third group contains the MQMessage, MQPutMessageOptions, and MQGetMessageOptions classes. They encapsulate the MQMD, MQPMO, and MQGMO data structures respectively, and are used to put and retrieve messages to/from queues.

The class relationship is presented in Figure 103.

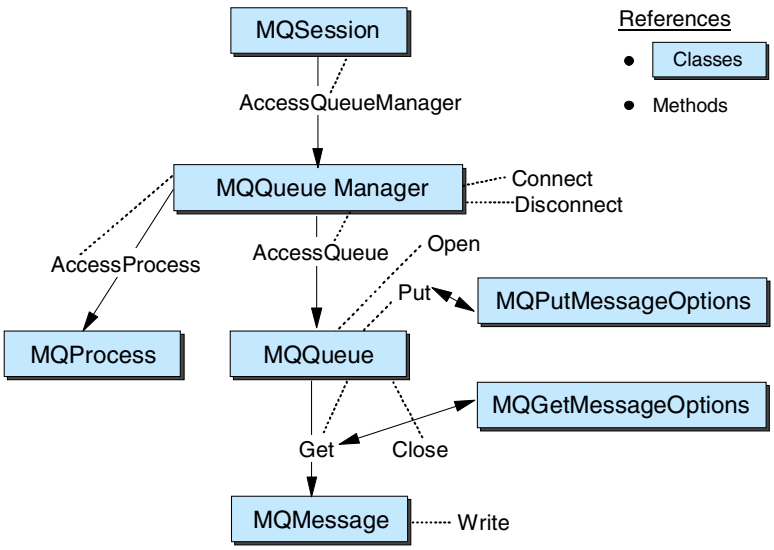


Figure 103. MQLSX class model

In reference to Figure 103, the following list will help identify the components:

- The MQSession class defines an instance of MQSeries access. This class manages all calls and error conditions. Its AccessQueueManager method is used to set the MQQueueManager class object.
- The MQQueueManager class is used to access the MQSeries Queue Manager (MQM). This is performed by the Connect method for access and the Disconnect method to indicate completion.
- The Disconnect method must be used in the newer version of the MQLSX for syncpoint control. Please ensure you code this statement.
- The MQProcess class object is set by the AccessProcess method of the MQQueueManager class. The MQProcess class enables you to read the MQSeries process definitions.
- The MQQueue class object is set by the AccessQueue method of the MQQueueManager class. The MQQueue class allows you to manipulate your queues. You must use the Open method to access your queue, and when you are done, use the Close method to close (commit) your work.
- The MQPutMessageOptions class, along with the Put method of the MQQueue class, define the complete put operation to the MQ Queue.
- The MQGetMessageOptions class, along with the Get method of the MQQueue class, define the complete get operation from the MQ Queue.
- The MQMessage class contains the Write and Read methods that enable you to manipulate the message buffer. This buffer is used by the Put and Get operations as the storage area for your message content.

9.2.2 A sample parts ordering application

We included a parts ordering database in the portal. We have chosen a scenario which is part of many procurement solutions. A supplier offers items on the Internet. Customers can order these items using a standard Web browser.

The customer can either be other departments inside of our company, if the company offers parts, or the application could be in another company that is connected to our company with this B2B application. Then we can access the application from the extranet when the application is running on the customers' extranet Web site.

In our scenario, each customer has an account where orders are managed. This is done by a transaction-based CICS application derived from the standard sample application FILEA. The application is illustrated in Figure 104 on page 191 and described in detail in the following text.

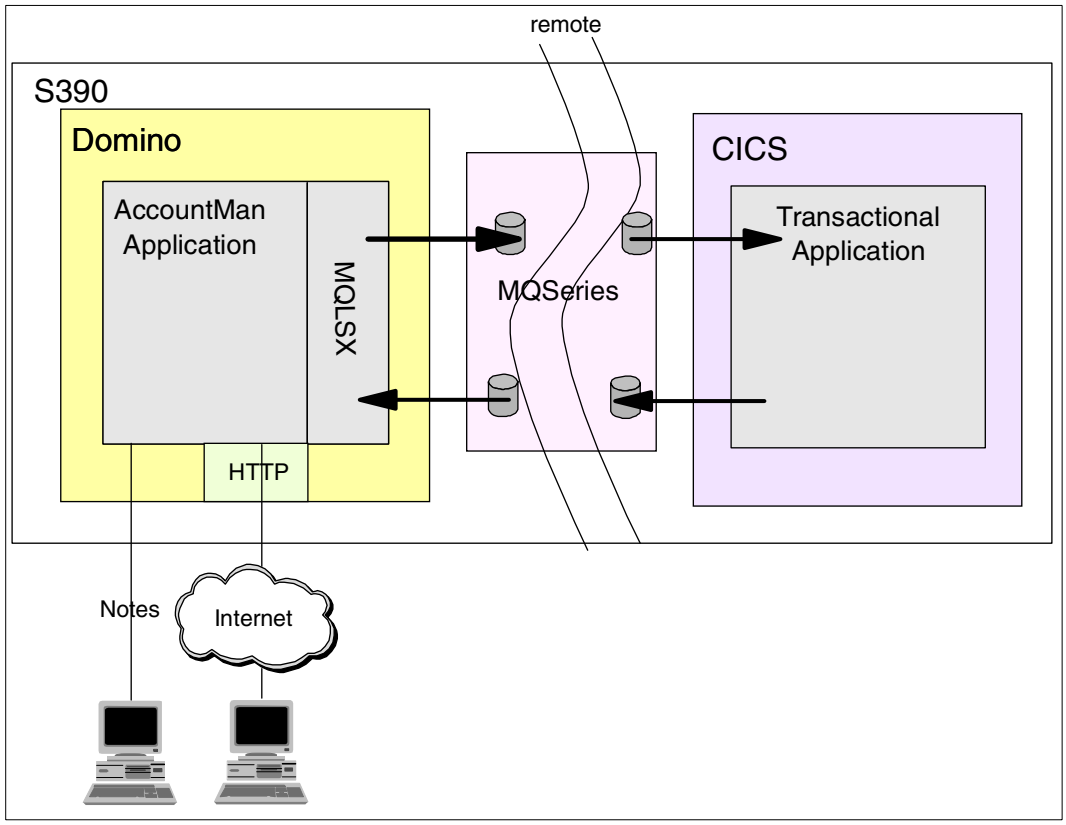


Figure 104. MQLSX order account management scenario

We developed a Domino application that can be accessed from the Web or from a Notes client. When an item is ordered, the order will be saved in the Domino server database using the Web browser; see Figure 105 on page 192.

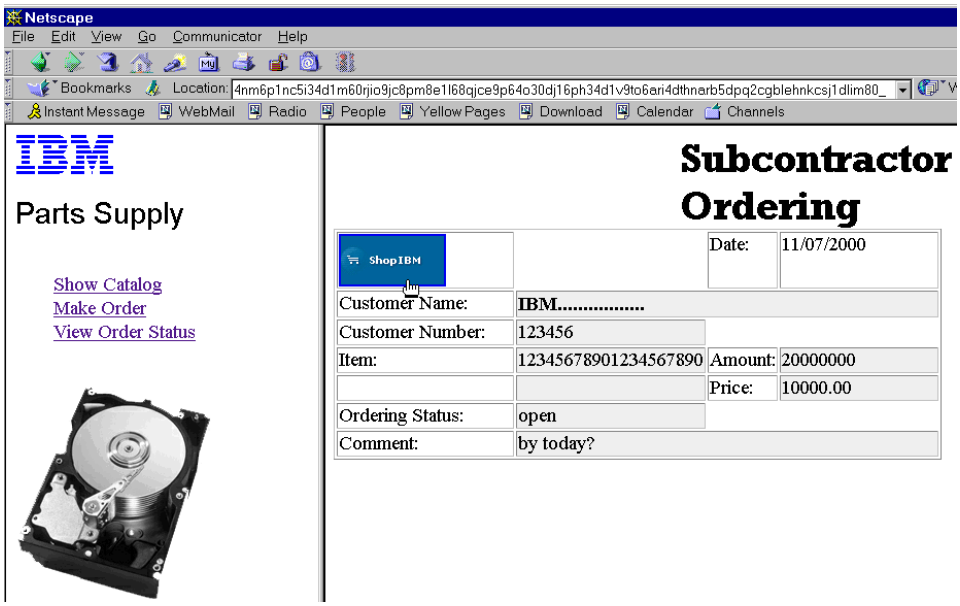


Figure 105. Putting an order into MQSeries

By clicking **Shop IBM**, a server-based agent is activated. It puts this order, in a defined format, into an MQ queue called SG245682.OUT.QUEUE. This is accomplished using the MQLSX classes. This message format contains the ID of the document representing the order, so the connection to the originating Notes document can always be obtained. If the action is successful, this message is received by the browser:

```
"Your Order is shipped. You will receive the confirmation soon."
```

MQSeries itself can be distributed in various ways. The MQ Manager handles the queue entry. If necessary, it distributes it using MQ transfer queues to a distributed server.

A CICS application reads the queue and gets the MQ message issued by the Domino server. Once the message is retrieved, the CICS transaction logic saves the account order information to a VSAM file. When it is saved, a confirmation is put into the SG245682.NOTES.IN.QUEUE, which is managed and possibly transferred to another location by the queue manager.

A scheduled Domino agent runs every five minutes to poll the SG245682.NOTES.IN.QUEUE queue. Of course this parameter is changeable. When the message arrives in the queue being polled, it is read by the agent. The ID of the corresponding Notes document is retrieved from

the string and the document is modified with the information issued by the CICS application. For example, confirmation information is shown in Figure 106.

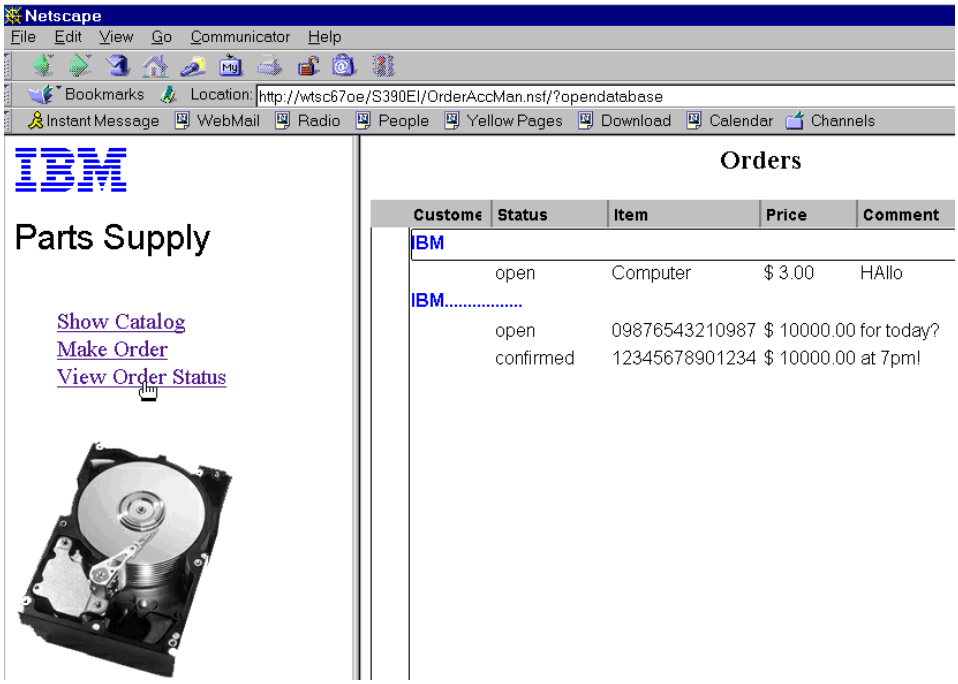


Figure 106. Viewing status change in the view after MQ get

The confirmation information is now visible in the document and the view. At this point, some automation (such as mail notifiers) could be added with little effort. Of course our application is very simple, since we concentrated on the queue interface management.

9.2.2.1 Portal integration of the application

The application can be accessed from the portal by switching to the TechSupport Portal Page. Clicking on links allows you to create a new parts order or to view the current ordering status, as shown in Figure 107 on page 194.

Figure 107. Embedding the parts ordering application in the portal

For detailed information about how the coding and configuration of this application is accomplished using Domino, MQSeries, and CICS, refer to *Lotus Domino for S/390 Release 5: Enterprise Integration Using Domino Connector Products*, SG24-5682.

9.3 Domino and Java

Domino R5 is a complete Web application server that fully supports the Java environment. Domino applications can be written in Java since you can call into the Domino object interface from a Java program. Domino also supports CORBA to build distributed Domino applications. Finally, Domino supports JDBC calls to allow Java programs access to Domino data (though not on S/390).

Domino R5 supports many Internet models of programming, so you can choose your favorite language when designing Web applications—whether that language is JavaScript, Java, HTML 4.0, or LotusScript.

Java and JavaScript support are both available within Domino Designer. Domino Designer R5 includes a Java editor and Java Virtual Machine (JVM) for developing applications. This allows you to create and compile Java agents, and edit all scripts and formulas, all from within Designer.

With native support for JavaScript and HTML in the Notes client, you can, in theory, design applications that run the same on the Web as they do within

Notes. In addition, Domino R5 allows you to use third-party design tools, such as NetObjects Fusion, NetObjects ScriptBuilder, and IBM VisualAge for Java.

From a Java program coded as an application, a Domino agent, an applet, or a servlet, you can call into the Domino object interface. In Domino R5, you need to import the lotus.domino package. The lotus.domino package has the same content as the R4.6 lotus.notes package, and supports the classes, methods, and enhancements of the new release.

The lotus.domino classes allow you to access named databases (lotus.notes.Databaseclass), views and folders (lotus.notes.Viewclass), documents (lotus.notes.Documentclass) within a database, and items within a document (lotus.notes.Itemand lotus.notes.RichTextItemclasses). The session class (lotus.notes.Sessionclass) is the root of the Domino object hierarchy, providing access to the other Domino objects, and represents the Domino environment of your Java program.

More information about Domino Enterprise Integration and Java can be found in *Lotus Domino for S/390 Release 5: Enterprise Integration Using Domino Connector Products*, SG24-5682.

Chapter 10. NNTP and the Domino server

News is a key element in a portal. News can take many forms. It could be just a source of information for the people in the company (enterprise news), or it could take the form of people asking for help, or posting ideas about a specific topic and receiving responses.

Newsgroups and newsfeeds are an excellent way to address the problem of providing people with information specific to their needs, or for them to get information from peers in a specific industry or field.

USENET news is one of the most popular services on the Internet. The public newsgroups provide discussion areas full of articles and messages on a vast array of topics—technical, recreational, controversial. Since its inception with RFC 977, NNTP as evolved as an information exchange for users with common interests—a virtual water-cooler in a cybernetic hallway. The number of USENET newsgroups is in the thousands, covers a multitude of subjects, and grows at a regular rate. The USENET newsgroups are distributed as newsfeeds using the NNTP protocol and NNTP servers.

10.1 USENET

USENET is a world-wide distributed discussion system. It consists of a set of newsgroups with names that are classified hierarchically by subject. Articles or messages are posted to these newsgroups by people on computers with the appropriate software. These articles are then broadcast to other interconnected computer systems via a wide variety of networks. USENET is available on a wide variety of computer systems and networks, but the bulk of modern USENET traffic is transported over either the Internet or UUCP.

USENET is the set of people who exchange articles tagged with one or more universally recognized labels, called newsgroups (or groups). Some newsgroups are moderated; the articles are first sent to a moderator for approval before appearing in the newsgroup.

There is often confusion about the precise set of newsgroups that constitute USENET. One commonly accepted definition is newsgroups listed in the periodic “List of Active Newsgroups” postings which appear regularly in news.lists.misc and other newsgroups. A broader definition of USENET would include the newsgroups listed in the article “Alternative Newsgroup Hierarchies” (frequently posted to news.lists.misc). An even broader definition includes newsgroups that are restricted to specific geographic

© Copyright IBM Corp. 2001

regions or organizations. Each USENET site makes its own decisions about the set of groups available to its users; this set differs from site to site.

Note

Note that the correct term is “newsgroups”; they are not called areas, bases, boards, bboards, conferences, round tables, SIGs, echoes, rooms or usergroups! Nor, as noted above, are they part of the Internet, though they may reach your site over it. Furthermore, the people who run the news systems are called news administrators, not sysops. According to USENET: If you want to be understood, be accurate.

For more information on USENET and a list of available newsgroups, visit: `http://sunsite.unc.edu/usenet-i`.

Ever since Domino 4.6, Network News Transfer Protocol (NNTP) has been an integrated part of the suite of Domino services for secure and interactive applications. Using the standard Internet NNTP protocol, the Domino NNTP server can participate in the distribution of public USENET news and private discussion groups over the Internet. Domino NNTP users can view either private discussion groups or public USENET newsgroups from Notes, from a standard news reader client or from a Web browser. The Domino server NNTP process is illustrated in Figure 108 on page 199.

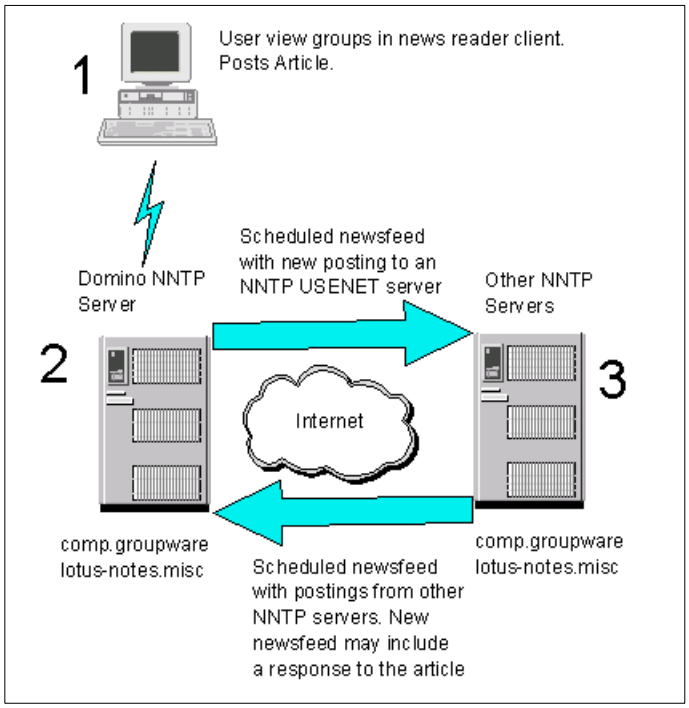


Figure 108. Domino server NNTP process

10.2 Advantages of using Domino for NNTP

Domino fully supports user participation in USENET newsgroup discussions and participates in the distribution of USENET news throughout the Internet.

Domino offers flexibility in newsreader clients, letting the user participate in newsgroup discussions by reading and posting articles from Notes, from a client news reader using the NNTP protocol, or from a Web browser. This is an advantage for organizations whose networks contain multiple platforms and clients. It also allows non-Notes members of a Notes organization to participate in newsgroup discussions through a standard NNTP newsreader or a Web browser.

Domino allows direct access to the NNTP server or indirect dial-in access by modem to a server that has direct access to the remote NNTP server.

Domino provides Internet Secure Sockets Layer (SSL) security for your server. Domino supports SSL version 3.0 authentication and encryption, as well as password authentication.

Domino supports private and moderated NNTP discussion groups with the familiar Notes discussion views. Private groups can be distributed within your intranet or can be more widely accessed by authenticated users across the Internet.

Domino also supports archiving and expiration for newsgroups and allows Notes clients or Web browser clients to create personal interest profiles for notification of news articles on specific topics.

Domino assures that fidelity of MIME-formatted news messages is preserved.

By using the NNTP server in Domino, you have the option of applying Notes document management to newsgroups. When the newsgroup is hosted on a Domino server, you have all the power of Notes at your disposal for disseminating the information: replication, access control, security, full text search, and the capability to create agents. One example of a mail agent would be to send yourself a doclink when the agent finds an article on certain subjects.

Domino NNTP takes advantage of the Domino object store to serve up newsgroups to both Notes clients and Internet newsreaders. One of the most powerful features of the Domino NNTP server is having a discussion database in a single object store. The information in the discussion can be accessed by multiple protocols and client types (NNTP, HTTP, Notes) simultaneously.

Using NNTP with Domino lets anyone using any of the three types of clients (Notes, Web browsers, or NNTP news readers) participate in the same discussion. For example, in a discussion of product support, contributors can post with Notes, or with the news readers or Web browsers included with Netscape Communicator or Microsoft Outlook. It is an excellent option for anything you want to publish on the Web because it allows a broad range of clients to access the discussion.

10.3 Domino NNTP security features

Perhaps the most important feature of the Domino NNTP server is the configurable security, which applies to all NNTP newsreaders and remote NNTP servers accessing your Domino server. You can configure all of the following security features on your Domino NNTP server:

- Secure Sockets Layer (SSL) Encryption and Authentication: Domino supports SSL encryption and accepts SSL client certificate authentication over an SSL port. Refer to Chapter 5, “Portal security” on page 77.

- User Authentication: Name and password authentication is configurable over the standard TCP port and over the SSL port. Users must log on to the Domino NNTP server using a username/password pair that is recognized by the server in the Public Address Book. Authenticated users are permitted to access newsgroups based on newsgroup ACLs.
- Anonymous Access: Domino can be configured to allow Anonymous user access over the standard TCP port or over the SSL port (or both). Anonymous users can access all newsgroups that permit Anonymous access in the ACL.
- Restricted Host Access: The Domino NNTP server can be configured to grant or to deny access by default to all NNTP clients or remote NNTP servers. Exceptions to the default access can be listed and can include wildcards. For example, you can *deny* access to all hosts except 111.222.333.*. This setting would only allow access to your server from machines on that subnet; or you can *grant* access to all hosts except 111.222.333.*, which would let all hosts access your server, except for machines on that subnet.
- Restricted Newsgroup Access: Using access control lists, you can restrict access to individual newsgroups. For example, when you create a private newsgroup, you can set the ACLs to allow internal users to read and post articles to the group (author access), while allowing external Internet users to read articles but not to post new articles (reader access).
- Restricted Newsgroup Posting: You can configure your Domino NNTP server to prohibit posting from all NNTP clients and to prevent receiving new articles from remote NNTP servers.

10.4 Managing the newsfeeds

Administrators must be able to manage the newsfeeds (the distribution of articles between NNTP servers). USENET newsgroups number in the thousands. Although many groups are technically-oriented and useful in the workplace, one may not want or need to access them all, as some may be controversial or offensive to participants at your site. You should also consider whether your network and other resources can handle the massive amounts of data in a USENET newsfeed.

Domino NNTP server lets you select the newsgroups you want and choose from the newsfeed type that complies with your organization's policies. Newsfeeds are configurable in the Connection document and the Server document. The convenient naming hierarchy facilitates the use of the wildcard for specifying groups. For example, many technical newsgroups

reside under the *comp* hierarchy. On a wildcard search, you could use comp.* or comp.os.*

For each newsfeed, you must select a newsfeed type that determines which server initiates the feed and in which direction the articles are transferred. You can select from the following types:

1. PULL feed: Your server initiates the connection, requests newly posted articles from the remote server, and stores the new articles.
2. PUSH feed: Your server initiates the connection and sends newly posted articles to the remote server.
3. ACCEPT feed: Your server accepts a connection from a remote NNTP server that sends newly posted articles to your server. An Accept feed is the opposite of a Push feed; one server pushes, the other accepts. This is the most common method of distributing USENET news.
4. PULL- PUSH feed: A combination of the PULL and PUSH feeds. Your server initiates a connection to a remote NNTP server, requests or pulls newly posted articles from the remote server and then sends (pushes) articles to the remote server that have been posted locally since the previous newsfeed.

An additional newsfeed type pulls from your Domino server. A remote NNTP server connects to your server and requests your server to send it newly-posted articles. The connection is defined on the remote server and is not a newsfeed type that you choose in the Connection document.

10.5 NNTP configuration

Setting up NNTP on the Domino server involves configuring the Server document that will act as newsfeeder and creating a connection document to connect to USENET groups.

To access USENET groups or to allow Internet users to access your private discussion groups, you must establish a connection with an Internet service provider (ISP).

If you are planning to participate in USENET newsgroups, consider the following issues:

- Which newsgroups do you want to send to or receive from?
- Do you have enough disk space to store the newsfeeds and archived articles? Conservative estimates put the daily USENET newsfeed distribution at 4 to 5 GB.

- What is your organization's policy on posting to public USENET newsgroups? Do you want an employee to post personal opinions under the company logo? Consider setting up a policy for postings from your organization and publish it within your organization.

You can use NNTP on your server to receive newsfeeds and connect to other newsgroups or to act as an NNTP server serving newsgroups.

10.5.1 Setting up the server to receive newsfeeds

To set up NNTP on our server to receive newsfeeds, we completed the following tasks:

1. Set up newsfeeds in the server Connection document.
2. Set up server and client security in the Server document.
3. Set up capacity and posting restrictions in the Server document.
4. Set up access control on individual database access control lists.

10.5.1.1 Newsfeed connection document

First, we set up the Connection document for connections to a remote NNTP server for the newsfeeds. We used IBM's "news.zurich.ibm.com" NNTP server for our needs. The detailed steps follow.

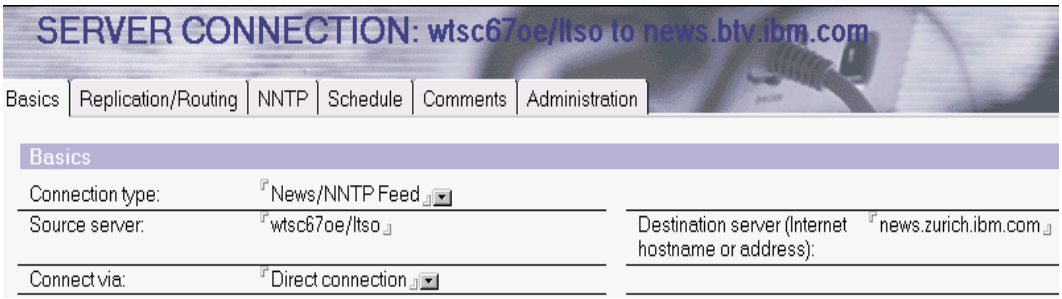


Figure 109. Setting up a server connection document for NNTP

1. Go to **Connections** in the Domino Directory and choose **Add Connection**. In the Notes R5 Administration Client you will find this under the Configurations tab.
2. Choose **News/NNTP** as the connection type.
3. Name your Domino NNTP server as the Source.
4. Choose **Direct** or **Dial-in** for the newsfeed. If you are connecting to the server locally, or have a direct internet link, choose **Direct**.

5. Name the destination server (the remote NNTP server). Enter its fully qualified Internet host name or IP address.

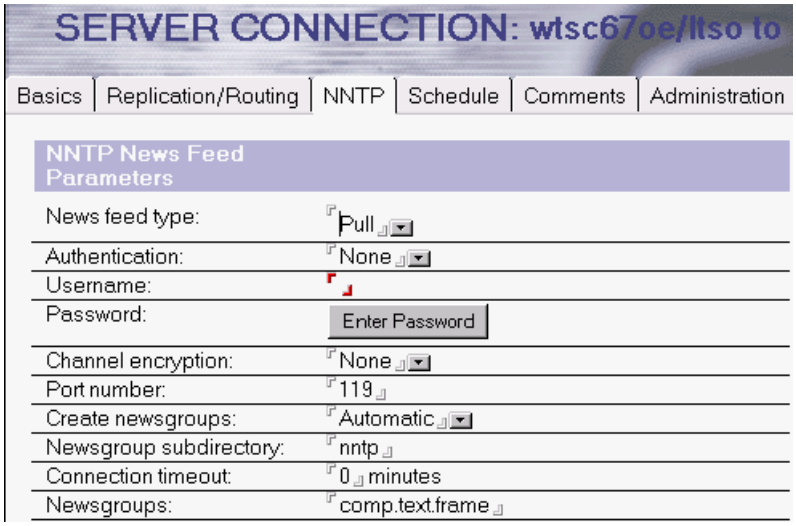


Figure 110. Setting up NNTP news feed parameters

6. In the NNTP Newsfeed parameters section:
   a. Select **Accept** to allow the *remote* server with NNTP to initiate a connection to the Domino server with NNTP and send new articles. In our case we used the **Pull** option so that we can manage the connection time and schedules for download.
   b. For Authentication, we used **None** (which is the default). Use **SSL** when the remote server with the NNTP service requires that the Domino server with NNTP use SSL. The SSL port must be **Enabled** on the **Port** -> **Internet Port** -> **News** tab on the Server document.
   c. In the Create newsgroups field choose **Automatic** (default) to create a newsgroup for each group that is included in this newsfeed and that does not yet exist on the server, or **Manual** to manually create all newsgroups on the server
   d. Enter the name of the subdirectory that will store newsgroups created during a newsfeed; the default is nntp.
   e. The maximum time, in minutes, that a newsfeed has to complete; the default is 0 minutes.
   f. In Newsgroups enter the names of any newsgroups that you will pull articles from or push them to. In this field you can use wildcards; for example, microsoft.access.*. We only wanted one newsgroup: comp.text.frame.

10.5.1.2 Security in the server document

Next, set up security in the server document for incoming client and server connections using the following steps:

1. Open the Server document for the server where NNTP is enabled.
2. Open the Internet Port and Security Configuration section to specify the port settings for incoming access to the News (NNTP) service. Default port numbers should be appropriate. Enable the TCP/IP and/or SSL ports for incoming connections and decide if you want to allow anonymous or authenticated access. If you require password authentication on TCP or SSL, make sure to add an Internet password in the Person document for the user/server.

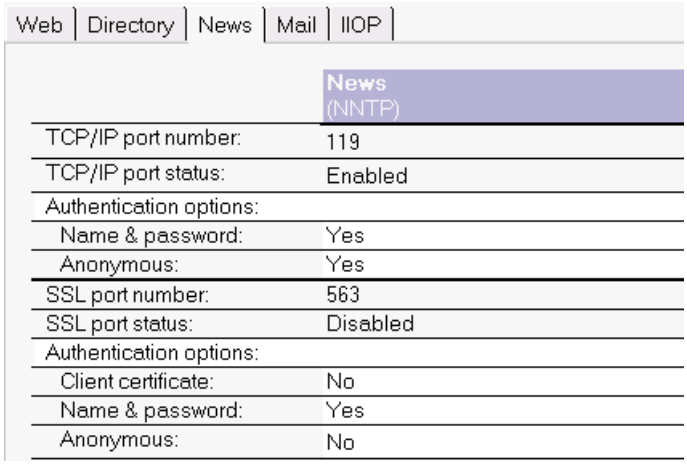


Figure 111. NNTP port security

10.5.1.3 Setting up capacity and newsgroup preferences

While the server document is still open, set up capacity and posting restrictions for the newsfeeds and newsreader clients, and choose other optional administrative tasks.

1. Open the NNTP Server section under Internet Protocols. In the Basics section, enter the NNTP administrator's Notes/Domino name, select a logging level, and configure host access.
2. In the Newsreader parameters section, select whether to allow posting, and set capacity and a maximum timeout for the news reader clients Defaults are provided for the maximum post sizes and maximum connection size for the newsreader client.

3. In the News Feed Parameters section, select whether to allow posting to your NNTP server during newsfeeds, set capacity for the postings, and the maximum timeout. Defaults are provided for the maximum posting sizes and maximum connection size for a news feed.

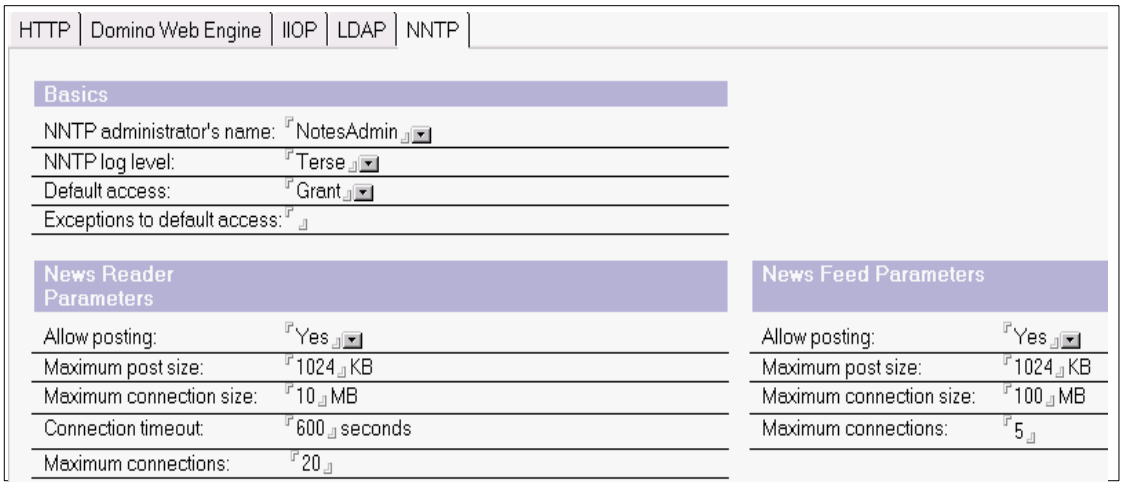


Figure 112. NNTP server preferences

10.5.1.4 Set the access control for specific newsgroups

Set the access control on individual newsgroup databases. To access newsgroups that do not allow anonymous access, users and servers must provide acceptable authentication to the server. Authentication can consist of a user name/password pair or an SSL client certificate that can be found in a Person document of the Public Address Book. The username and password pair allow Domino to provide access to the server and to specific newsgroups.

10.5.1.5 Viewing the newsgroup

After the pull of the newsgroup is complete, you are able to open the automatically-created database named after the newsgroup: comp.text.frame. It is available in the NNTP directory, as specified in the connection document.

When opened, the database looks like Figure 113 on page 207. This database can now be replicated to other Domino servers. Users can view the news articles. Should they want to respond to some articles, your connection document connect type must be changed to pull-push (refer to Figure 114 on page 209) so that responses can be replicated back to the newsgroup.

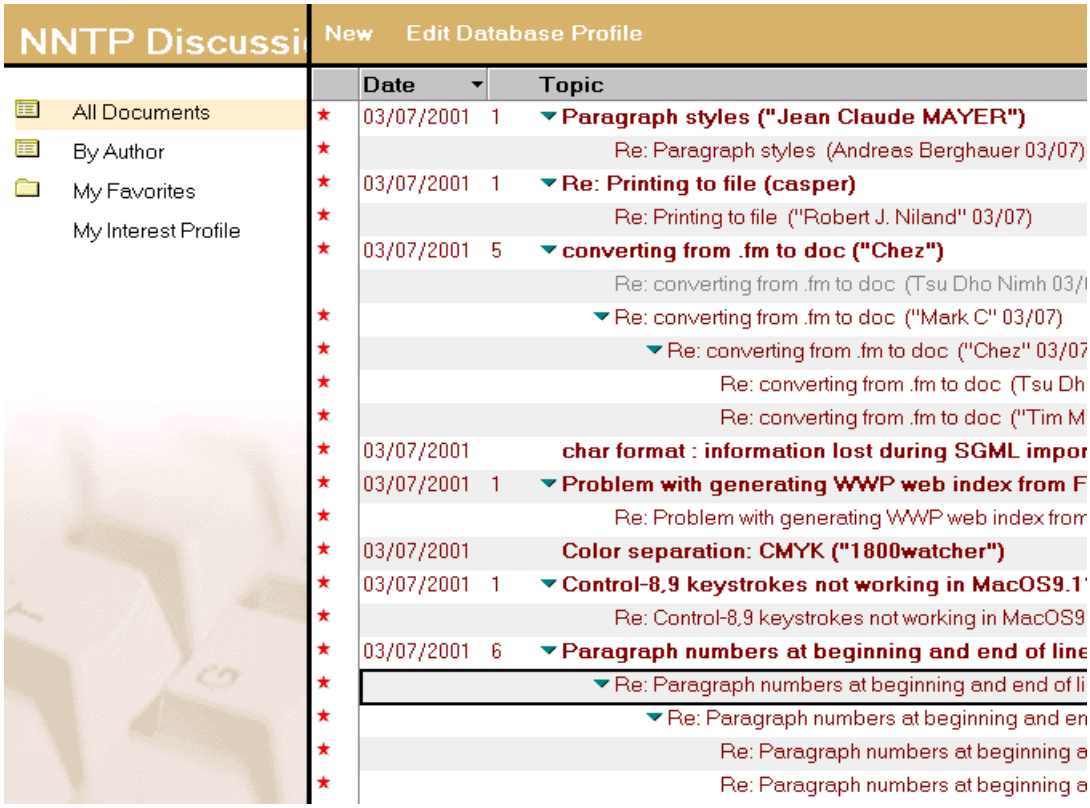

NNTP Discussi
New
Edit Database Profile
All Documents
By Author
My Favorites
My Interest Profile
Date
Topic
03/07/2001 1 Paragraph styles ("Jean Claude MAYER")
Re: Paragraph styles (Andreas Berghauer 03/07)
03/07/2001 1 Re: Printing to file (casper)
Re: Printing to file ("Robert J. Niland" 03/07)
03/07/2001 5 converting from .fm to doc ("Chez")
Re: converting from .fm to doc (Tsu Dho Nimh 03/
Re: converting from .fm to doc ("Mark C" 03/07)
Re: converting from .fm to doc ("Chez" 03/07
Re: converting from .fm to doc (Tsu Dh
Re: converting from .fm to doc ("Tim M
03/07/2001 char format : information lost during SGML impor
03/07/2001 1 Problem with generating WWP web index from F
Re: Problem with generating WWP web index from
03/07/2001 Color separation: CMYK ("1800watcher")
03/07/2001 1 Control-8,9 keystrokes not working in MacOS9.1
Re: Control-8,9 keystrokes not working in MacOS9
03/07/2001 6 Paragraph numbers at beginning and end of line
Re: Paragraph numbers at beginning and end of li
Re: Paragraph numbers at beginning and en
Re: Paragraph numbers at beginning a
Re: Paragraph numbers at beginning a

Figure 113. Automatic newsfeed downloaded from a USENET server.

10.5.1.6 Considerations when pulling from newsgroups

We had some trouble getting the PULL to work. Here are a few pointers to keep in mind:

1. Do you have authorization to access to the newsgroup? It might be necessary to get permission and an ID/password from the remote USENET group server before the connection will work.
2. Is your server available on the Internet (that is, located in the demilitarized zone or DMZ) and correctly defined in the DNS?
3. Is port 119 open on your firewall to allow NNTP on that port?
4. It might be a good idea to put your Domino NNTP server in the DMZ and allow it to replicate newsgroups to internal Domino servers. Servers located in the DMZ must also be defined on Internet DNS servers.
5. Consider dedicating a Domino server for NNTP pulls and pushes. We used just one newsgroup for testing purposes and found that our server

was totally unavailable to other users for the duration of the download process.

10.5.2 Setting up to serve newsgroups

You can set up your Domino server to serve newsfeeds to your local users or Internet users. If you want to allow Internet users to receive news from your Domino NNTP server, you must establish a connection with an Internet service provider (ISP). Steps to follow to set up your server as a newsfeeder include some of the same functionality previously described for receiving news from USENET groups:

1. Create a Newsgroup.
2. Set up access control on individual database access control lists.
3. Set up server and client security in the Server document.
4. Set up capacity and posting restrictions in the Server document.

For steps 3 and 4, refer to 10.5.1.2, “Security in the server document” on page 205 and 10.5.1.3, “Setting up capacity and newsgroup preferences” on page 205.

If you want to allow other servers to read from your newsgroups, include a connection document as described in 10.5.1.1, “Newsfeed connection document” on page 203, with the exception that the newsfeed type would be one of the push-pull combinations.

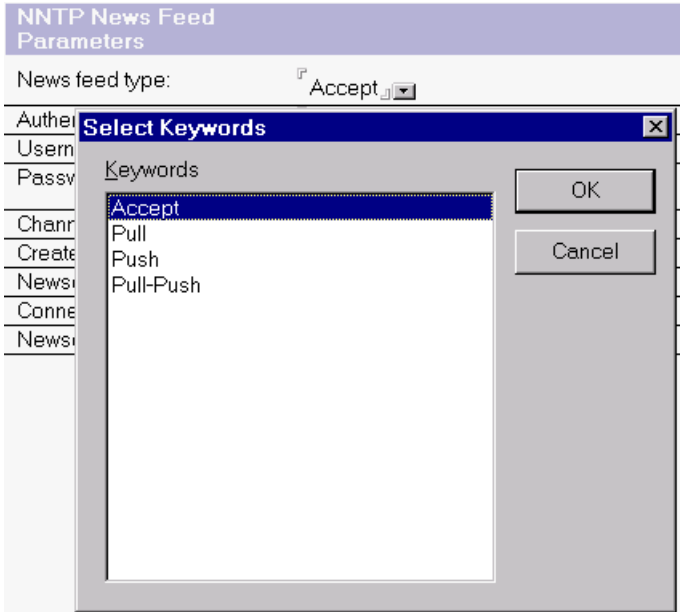


Figure 114. Newsfeed types

10.5.2.1 Creating newsgroups

Newsgroups can be created automatically during a newsfeed (refer to 10.5.1, “Setting up the server to receive newsfeeds” on page 203). Or they can be created manually with a Notes database using the NNTP discussion template nntpcl5.ntf and the following steps:

1. Select **File** -> **Database** -> **New**.
2. Select the server where you have NNTP enabled.
3. Fill in the database name and filename. The database name cannot contain spaces and should reflect a hierarchy of your newsgroups, for example itso.internal.management.
4. Select the template server, click **Show advanced templates** and select the **NNTP Discussion** template (nntpcl5.ntf).
5. Click **OK** to create the database.

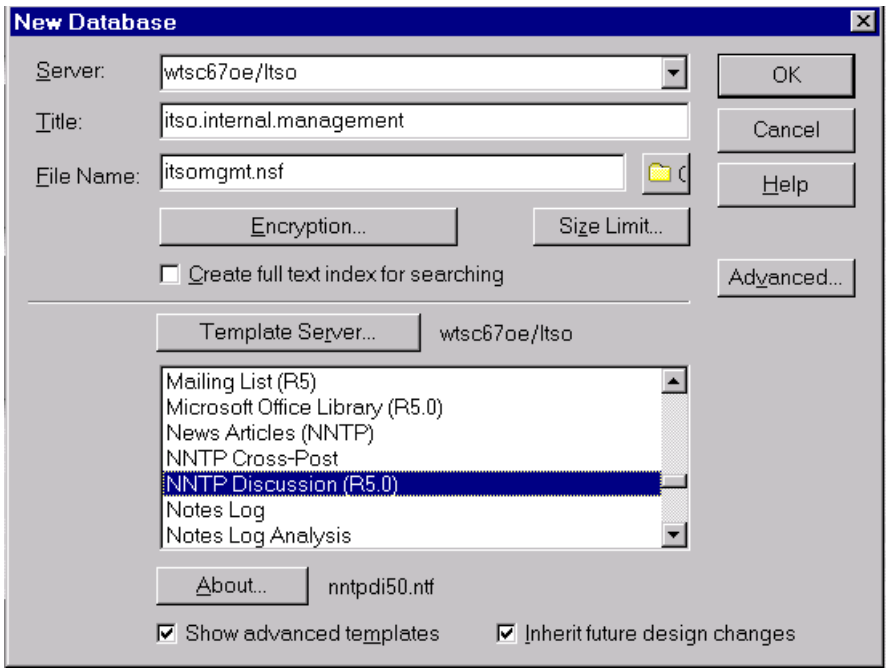


Figure 115. Creating a newsgroup

10.5.2.2 Editing the NNTP database profile

When you first open the newly created newsgroup, you will be prompted to edit the database profile document (refer to Figure 116 on page 211).

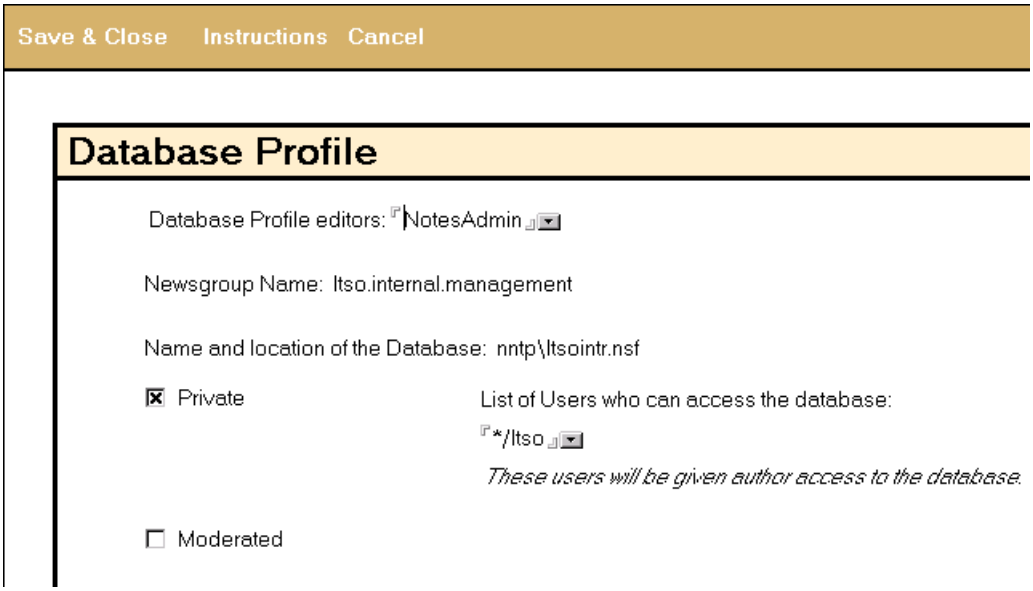


Figure 116. NNTP database profile

Each NNTP discussion database contains a database profile, which specifies information about the NNTP discussion. The newsgroup name is displayed and may be edited when the database profile is first created. Note that the newsgroup name is not editable for USENET newsgroups. Additional information includes whether the database is a private newsgroup (not a USENET newsgroup) or whether the newsgroup is moderated. Users who are invited to participate in the private newsgroup can be entered into the database profile and will be added to the database ACL with Author access. If the newsgroup is moderated, the moderator's name will be added to the ACL as Editor. The moderator's e-mail address needs to be included since would-be contributors will need to send the editor articles for approval.

Since this is an internal NNTP database, we changed the settings to private (which removes the default access of author from the ACL) and allowed author access to */ITSO (all ITSO Notes users will have author access).

The profile can be edited again at a later stage by clicking **Edit Database Profile** in the All documents view. Note that only users defined as Profile editors will be able to do this.

10.5.2.3 Set up access control on individual newsgroup databases

You have several options when you create a private newsgroup, which can be posted to the Internet to be accessed by authenticated users. The private

newsgroup can have access control that allows Web browsers, Notes users, and news reader clients to share in the same Notes database.

The access control list should reflect a group of users who may read and/or read and write documents. If this database is available to users outside your organization you should include either a default and/or anonymous access (reader or author), or only create external users to authenticate via name and passwords defined in person documents in the Name and address book. For more information regarding setting up security refer to Chapter 5, “Portal security” on page 77.

10.5.2.4 Viewing the news

Newsgroup users can view the news in three ways:

- From Notes: Simply open the database of any newsgroup that has been loaded or created onto the Domino server.
- From an NNTP news reader client: Select your Domino server as the news server or discussion server. Standard NNTP news reader clients are included with Netscape Communicator and Microsoft Internet Explorer.You will have to know the name of your news server (in our case wtsc67oe.itso.ibm.com) and complete any configuration requirements for the news reader client.
- From a Web browser using HTTP: Enter the URL of the Internet hostname of the server (for example, wtsc67oe.itso.ibm.com) and select from the list of databases on the server or specify the database to open the standard discussion view. From the browser's location field, enter the URL. Your administrator can help you determine the exact URL. If you create a private newsgroup on a server, you will have to distribute the URL to those users you are inviting to the newsgroup.

  The URL syntax typically has the form:

  ```
  http://domino_server/news_database.nsf/($All)?OpenView
  ```

  which would look like this:

  ```
  http://wtsc67oe.itso.ibm.com/nntp/itsomgmt.nsf/($all)?Openview
  ```

10.5.3 Integrate newsgroups into the portal

We integrated our newsgroup into the portal database by adding an outline entry to the Enterprise Outline. Depending on the focus of the newsgroup, you might add it to the functional menu (profiled according to roles), or to the enterprise menu. We added our itso.internal.management newsgroup to the enterprise menu (refer to Figure 117 on page 213).

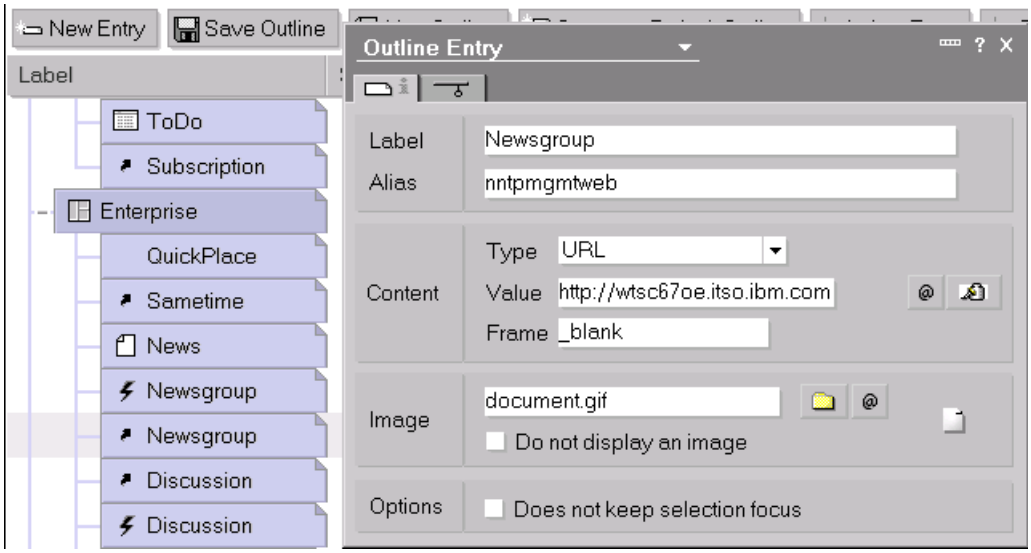


Figure 117. Adding newsgroup to enterprise outline menu

When you click on the Enterprise menu, Newsgroup entry, you will be presented with the view in Figure 118 on page 214, which will allow you to add documents to the database or respond to current documents.

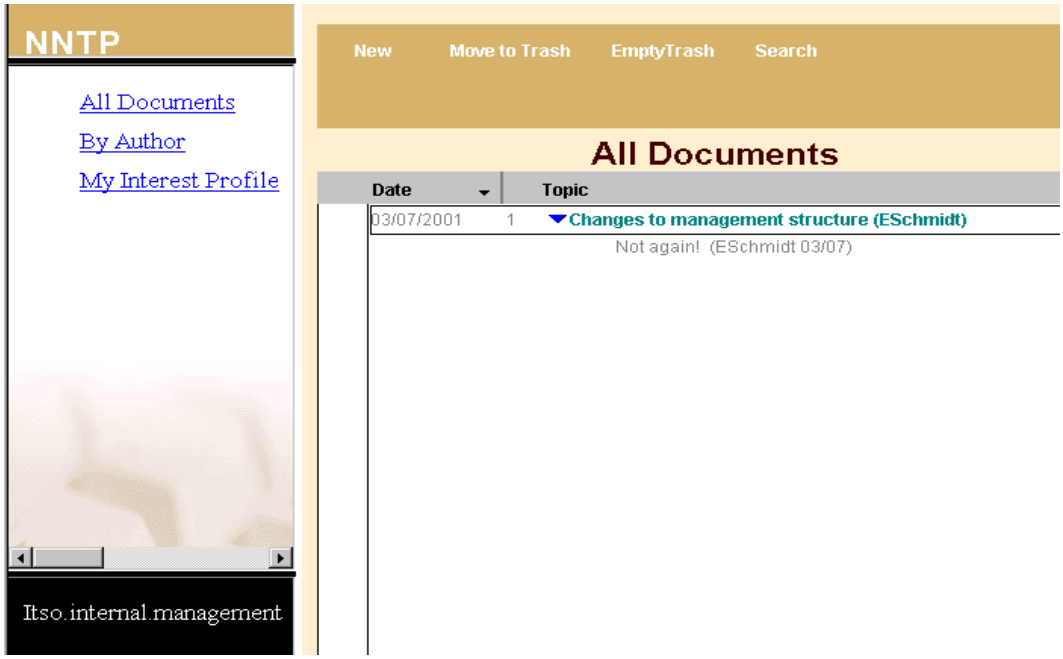


Figure 118. Opening the newsgroup database from a browser

Users can also set up a personal interest profile (refer to Figure 119 on page 215). Select the **Edit Profile** hotspot from any of the views in the database to bring up your personal interest profile document. You can elect to be notified in a newsletter if certain conditions are met. These conditions can include the appearance of your name or phrases that you designate. This feature is not available to Anonymous Web users or newsreader users. You will receive mail informing you of new additions to the database according to your profile.

.

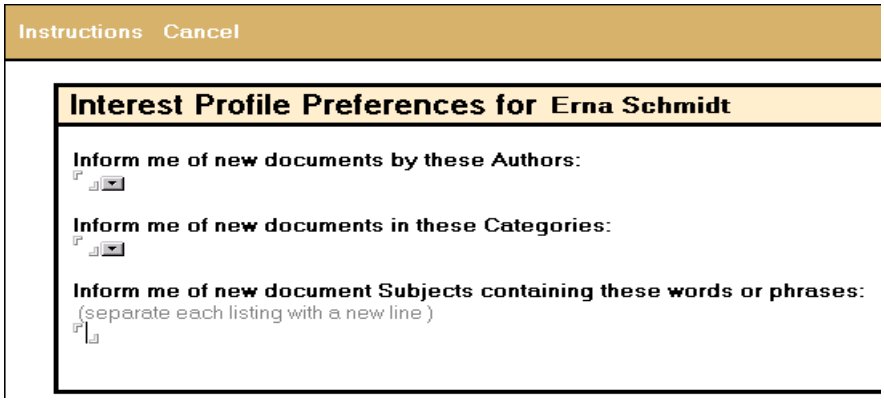

Instructions Cancel
Interest Profile Preferences for Erna Schmidt
Inform me of new documents by these Authors:
Inform me of new documents in these Categories:
Inform me of new document Subjects containing these words or phrases:
(separate each listing with a new line)

Figure 119. Setting up an interest profile

Chapter 11. Collaboration

This chapter describes how to implement Sametime and QuickPlace, two Lotus products designed to facilitate interaction and collaboration.

11.1 Sametime

This section describes how to add a Sametime server to the ITSO Domino domain, using a separate Windows 2000 server.

It also shows how to implement a Sametime Meeting Center page and a Sametime Chat applet in the portal database.

11.1.1 Sametime overview

Sametime consists of client and server applications that enable a community of users to collaborate in real-time online meetings. These collaborative activities are presence, chat, screen sharing, and a shared whiteboard that enable colleagues to meet, converse, and work together in instant or scheduled meetings. The Sametime 2.0 release also includes an optional Sametime Multimedia Services add-on package that brings real-time IP audio/video capabilities to Sametime meetings.

Sametime supports a broadcast technology that enables a large number of view-only users to watch a small number of users interacting during a meeting.

Each Sametime server contains a Domino Directory that maintains information about all users and servers that comprise the Sametime community.

The Sametime server task provides services to the Sametime client applications, including Community Services, Meeting Services, Broadcast Services, and Domino/Web Application Services. The Sametime Multimedia Services add-on provides Audio/Video Services.

The Sametime server is administered with the Web-based Sametime Administration Tool, available from the Administer the Server link on the Sametime server Web home page.

Additional information about Sametime is available at:

`http://www.lotus.com/sametime`

© Copyright IBM Corp. 2001

11.1.2 Installing the Sametime server on Windows 2000

Before installing the Sametime server, you must decide in which environment to install the server (Domino or Web-only), and what functions you want on your Sametime server (just the core Sametime server or the Multimedia Services add-on).

Our ITSO organization already uses S/390 Domino server Release 5.0.6a. We wanted the Sametime server to interoperate as a member of the existing Domino community (domain), so we chose to install the core Sametime server in a Domino environment. In this case, users access the Sametime server with a Web browser or the Sametime Connect client.

1. Register the Sametime server in the itso domain. As with a Domino server, we first register the Sametime server to add it to the itso Domino domain.
   a. Open the Domino Administrator client application, and select the server to be administered, wtsc67oe/itso.
   b. Select the **Configuration** tab, and choose **Server...** in the **Registration Tools** list.
   c. On the **Basics** panel, enter the server name, no password, the Domino Domain, and the Administrator name.
   d. On the **Other** panel, enter the Domino server title, the Notes Network name (this can be found in the wtsc67oe/itso Server document, Ports tab), the Local Administrator name and choose to store the server ID in a file.
   e. Click the **Register** button. This server is added to the Domain Directory. The server ID file will be used during the Sametime server setup. Be sure you can access it from the Sametime server or copy it to a floppy disk.
2. Install the Sametime server application on the server.
   a. Execute the setup.exe program, located in the **English -> Server installation** package directory.
   b. On the main dialog, click **Install the Sametime Server**.
   c. At the next dialog, Sametime Server Installation, you are prompted to choose between the **Sametime Core** or the **Complete** option: we chose the **Sametime Core** option.
   d. Read, follow directions, and click **Next** through the following screens:
      - Choose Installation Path
      - Start Copying Files

e. At the next dialog, Installation Complete, you should choose **Yes, I want to restart my computer now**. All files have been downloaded, but the setup is not complete. Click **Finish** to continue.

Note

Sametime is based on Domino R5.0.3 and later. It cannot run using a Directory design earlier than 5.0.3.

3. Set up the Sametime server.
   a. Enter the information in the appropriate fields.
   b. Browse to the appropriate network drive and directory to select the server ID. The server ID will be copied. When selected, the Sametime server ID should display automatically in the Sametime Server ID file field.
   c. Enter the name of the Domino server from which you want to pull (replicate) the Domino Directory (address book): `wtsc67oe/itso`
   d. Click **Next**. A progress bar displays and the installation program completes the setup; a Congratulations dialog displays.
   e. Click **Next**. The Sametime server setup progress bar displays; when done, the Setup is finished dialog displays.
   f. Click **OK** to start the Sametime services. The Sametime server setup progress bar appears. It might take two or three minutes for the services to start.
4. Set the home Sametime server for users in the Sametime community. The “home” server is the Sametime server to which each user connects for awareness and chat functionality. A user’s Community Services preferences, buddy lists, and other data are stored on the user’s home Sametime server.

   Each Person document in the Directory on the Sametime server contains a Sametime server field. The Sametime server field on each user’s Person document must contain the name of the Sametime server. You can create a simple agent to add the name of the Sametime server to the Sametime server field on each user’s Person document.

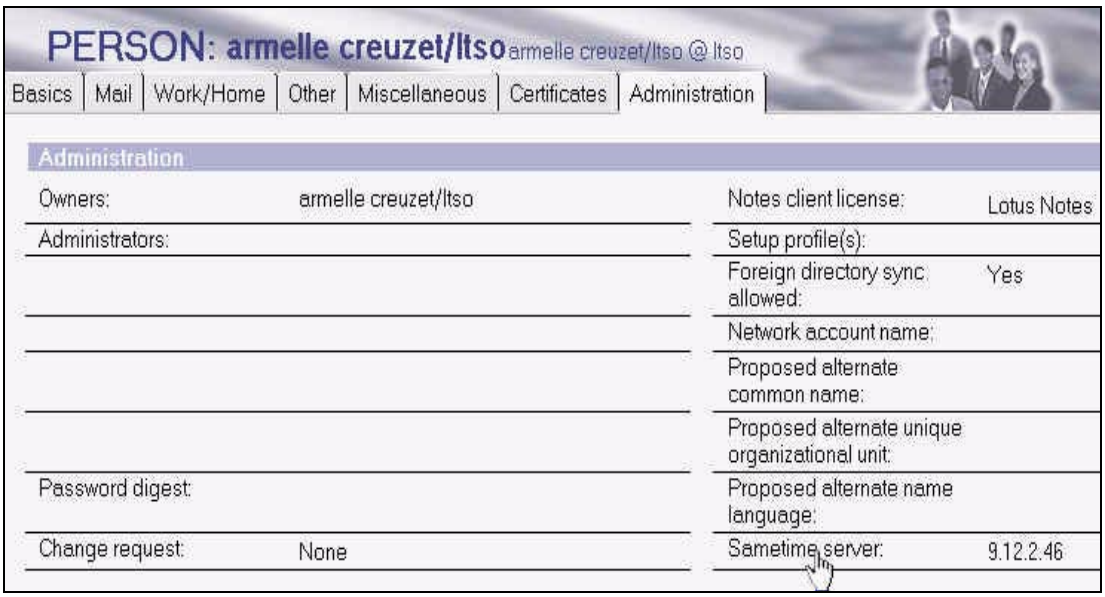


Figure 120. Adding the server to a Person document.

5. Validate the Sametime installation (dedicated server). Once the Server is set up, use the following procedures to ensure that Sametime is functioning properly:

   a. Validate Creating a New Meeting.

   a. Start the new meeting.

11.1.3 Integrating Sametime center into the portal database

The Sametime Meeting Center HomePage can be integrated into the portal database by adding an outline entry to the Enterprise outline. Refer to Figure 121 on page 221.

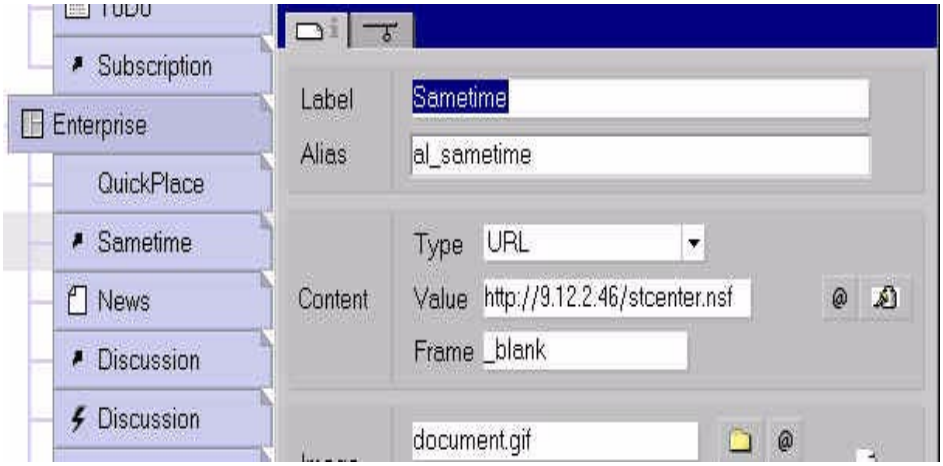


Figure 121. Adding an outline entry for the Sametime meeting Web page.

The Stcenter.nsf notes database, located in the Sametime server file system, must be entered as the home page in the Sametime server document.

Edit the Sametime server document, and select the **Internet Protocols** -> **HTTP** tab. Then enter `stcenter.nsf` in the **MAPPING** section, **Home URL** field, as shown in Figure 122.

Figure 122. Entering the home URL field

11.1.4 Using the Sametime Java applets

The Sametime server includes a list of sample Java applets which a developer can use to customize his environment.

One of these, the buddy list applet (LiveNames) displays a list of users that is read from an HTML parameter tag, along with their online status. An instant meeting can be initiated by right-clicking a selection of users and choosing the meeting type.

For more information about the code, refer to the white paper, “What’s New in the Sametime 2.0 Java Toolkit?”, and the tutorial PDF file included with the Sametime server.

To use this applet, create a new LiveNamesApplet.java file containing the Java sample source code provided in Appendix A.5.3, “LiveNamesApplet Java code” on page 259. Compile it; for example, use the javac program provided with the Java Development Kit (JDK).

JDK is available on the Sun Web site for Windows platforms, and on the IBM Java Web site for IBM platforms (AIX, OS/2, OS/390 UNIX System Services, OS/400, and VM/ESA). Each release of the JDK contains the following: the Java compiler, JVM, Java class libraries, Java applet viewer, Java debugger, and other tools. For the purpose of this redbook we used JDK 1.1.8.

The command `javac LiveNamesApplet.java` creates the LiveNamesApplet.class. This class is used by the LiveNamesApplet applet program. For more information about Java programming and components, refer to *Connecting Domino to the Enterprise Using Java,* SG24-5425.

The Sametime 2.0 Development toolkit is also needed to provide LiveNamesApplet with the classes necessary to run. The toolkit can be downloaded from the Sametime Meeting Center HomePage. Open the ntcenter.nsf database with a browser, and click the **Toolkit** link, located at the bottom of the page, and then download the ST20Java.zip. Unzip the package in a directory known by the local CLASSPATH environment variable, where the class will be compiled.

In Domino Release 5 applications, developers have the ability to store applets and code inside their application database. For Sametime-enabled applications where Java applets are used and the application is accessed via a Notes client only, developers can take advantage of this. If the application is accessed through a Web browser, the applets must be stored on the Sametime server file system. In this case you load applets from the Sametime server into your browser.

There may be policy/procedural/testing issues within your organization regarding putting files on the Sametime server; those must also be followed.

11.1.4.1 Creating an application database on a Sametime server

1. The first operation is to create a new database on the Sametime server.
   - Select from a Notes client menu: **File** -> **Database** -> **New**
   - Create this database on the Sametime server, and give it a name, such as `STPortal.nsf`. Refer to Figure 123 on page 223.

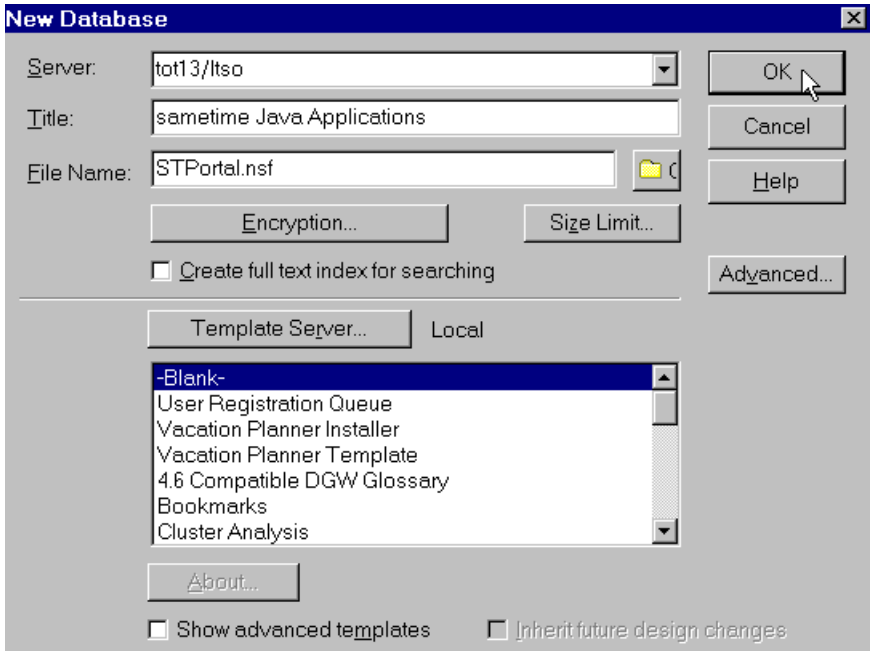


Figure 123. Creating a new Sametime application database

2. Once created, open both this new database with Notes Designer, and also the PortalS/390 database.
3. From the PortalS/390 database, copy and paste to the STPortal database the elements of design listed below:
   - Frameset EnterprisePortal
   - Frameset Main Portal
   - Page EnterpriseBody
   - Form MainMenu

11.1.4.2 Create the LiveNamesApplet pages

We first created one form containing the LiveNamesApplet inside the form. With this method, we had to code the login username and password as parameters to the applet. This worked well, but only one user could work with the applet, so we decided to provide a more sophisticated solution. We created this solution with the help of frames within a frameset.

To create the LiveNamesApplet pages, follow these steps:

1. In the Designer, create a new frameset, consisting of two frames. We named our frameset SameTime.

2. Name the one frame frmHidden and set its height attribute to zero percentage. This frame can not be seen by the user and is only provided to contain a form which will hold the user's username and password. The frame properties must be set to contain a named element of type form and we called the form Hidden.
3. Name the other frame frmSameTime and specify the frame properties to contain a named element of type form. We called the form Login.
4. Create the Hidden form. Add the following javascript to the HTML Head Content of this form:

```
"<script language=\'javaScript\'>" +
"var userName = \'A\';" +
"var password;" +
"</script>"
```

5. Create the Login form. Add two fields to this form. We named the fields txtUserName and txtPassword. Also create a hotspot button for this form. We called the button Login. The purpose of this form is to supply a method for the user to supply a username and password for the Sametime server.

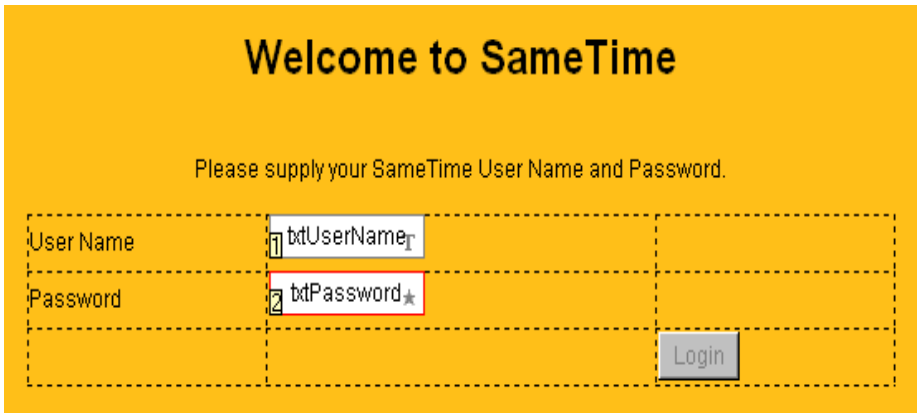


Figure 124. Sametime Login form

6. Add the following Javascript code to the *onLoad* event of the form to ensure that the UserName field gets the focus when the form is loaded:

```
document.forms[0].txtUserName.focus();
document.forms[0].txtUserName.select();
```

7. Add the following Javascript code to the *onSubmit* Javascript event of the form to ensure that both the username and password are supplied by the user and also to save the supplied data in the hidden form. If both the username and password are supplied, the code also loads another document into the frame. We named this form SameTime.

```
var returnValue;
returnValue = true;
if (document.forms[0].txtUserName.value == "")
{
    alert("You must supply a User Name.");
    document.forms[0].txtUserName.focus();
    document.forms[0].txtUserName.select();
    returnValue = false;
}
if (returnValue == true && document.forms[0].txtPassword.value == "")
{
    alert("You must supply a Password.");
    document.forms[0].txtPassword.focus();
    document.forms[0].txtPassword.select();
    returnValue = false;
}
if (returnValue == true)
{
    top.frames[0].userName = document.forms[0].txtUserName.value;
    top.frames[0].password = document.forms[0].txtPassword.value;
    document.location = "SameTime?OpenForm";
}
return false;
```

8. Create another form. We called this form SameTime. The purpose of this form is to display the LiveNamesApplet Java applet. Add the following code to the HTML Head Content of the form. The purpose of the code is to load the applet, supplying the username and password that were saved in the hidden frame, as parameters to the applet.

```
"<script>" +
"document.writeln(\"<APPLET WIDTH=200 HEIGHT=200 ID=LiveNamesApplet
CODEBASE=\'/STPortal.nsf/32f1d8e3611dee6a85256a09005e6910/$FILE\'
CODE=\'LiveNamesApplet.class\' ARCHIVE=\'CommRes.jar,STComm20.jar\'>\");" +
"document.writeln(\"<PARAM NAME=\'archive\' VALUE=\'STComm20.jar, CommRes.jar\'>\");"
+
"document.writeln(\"<PARAM NAME=\'cabinets\' VALUE=\'STComm20.cab,
CommRes.cab\'>\");" +
"document.writeln(\"<PARAM NAME=\'loginName\' VALUE=\'\" + top.frames[0].userName +
\"\'>\");" +
"document.writeln(\"<PARAM NAME=\'password\' VALUE=\'\" + top.frames[0].password +
\"\'>\");" +
"document.writeln(\"<PARAM NAME=\'watchedNames\' VALUE=\'Armelle creuzet,Allan
Smith,Ernesto Ramos,Tonie Erasmus,Erna Schmidt\'>\");" +
"document.writeln(\"<PARAM NAME=\'stserver\' VALUE=\'9.12.2.46\'>\");" +
"document.writeln(\"<PARAM NAME=\'cabbase\' VALUE=\'CommRes.cab\'>\");" +
"document.writeln(\"<PARAM NAME=\'cabinets\' VALUE=\'CommRes.cab,STComm20.cab\'>\");"
+
"document.writeln(\"</APPLET>\");" +
"</script>"
```

This concludes the creation of the pages for the LiveNamesApplet and provided a solution where the user can specify a username and password. With this solution we were able to use the full functionality of the LiveNamesApplet with different users logging in and using the Sametime functions.

11.1.5 Integrating the Sametime chat application into the portal database

The Sametime LiveNamesApplet is integrated to the portal database by adding two outline entries to the Personal Outline.

- The first outline entry is only for the Notes users, it is hidden from Web browsers.

  ```
  Label : Sametime Chat
  type : URL
  Value : http://sametimeserver/STPortal.nsf/SameTime?OpenFrameset
  ```

  “sametimeserver” is the name of the sametime server.

- The second outline entry is only for the Web browsers users, hidden from Notes 4.6 or later users.

  ```
  Label : Sametime Chat
  type : URL
  Value : http://sametimeserver/STPortal.nsf/SameTime?OpenFrameset
  ```

  “sametimeserver” is the sametime server name.

Figure 125 on page 227, shows an example of a Sametime chat obtained when clicking on the Portal database **Sametime Chat** menu entry.

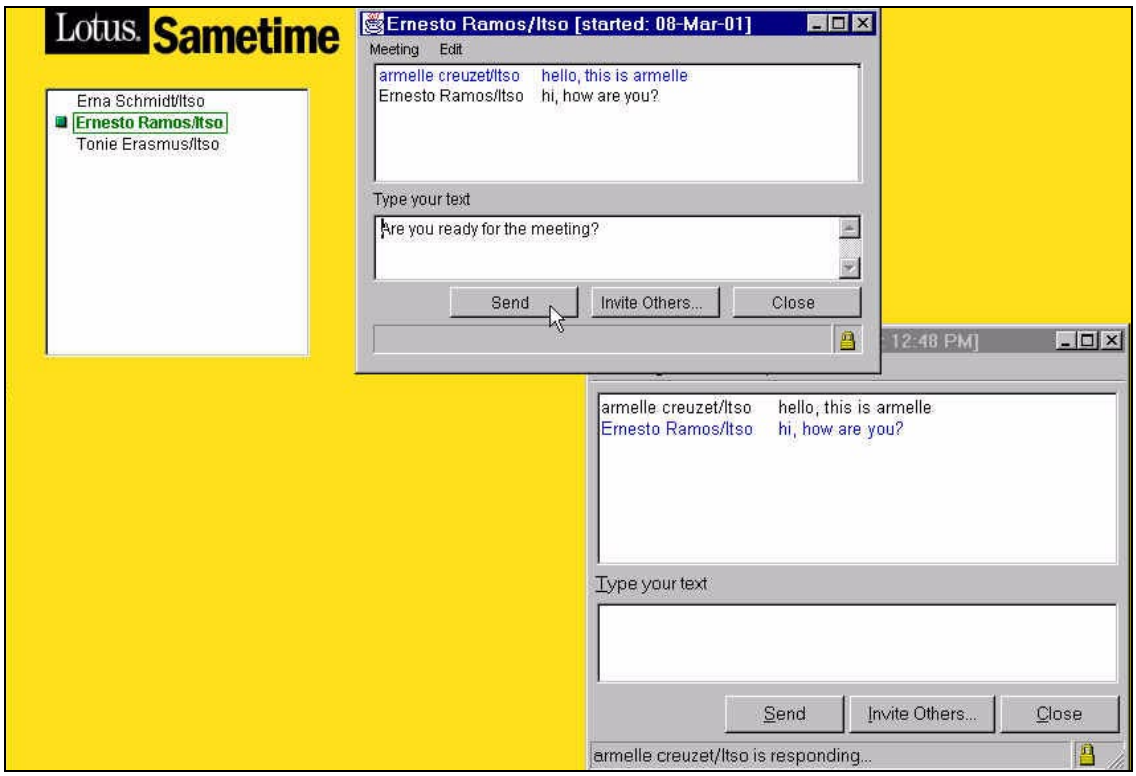


Figure 125. Accessing the Sametime community

11.2 QuickPlace

QuickPlace is a self-service Web tool for team collaboration. QuickPlace enables the instant creation of a team workspace on the Web. Teams use QuickPlace to share and organize ideas, content, and tasks associated with any project or ad hoc initiative.

11.2.1 QuickPlace overview

QuickPlace is a teamware solution. As an emerging category of software, International Data Corp (IDC) defines the attributes of *teamware* as follows:

- Primarily designed for smaller groups or teams
- Duration of usage is often limited or for a specific period of time
- Easy to install, set up, and get started
- Requires minimal infrastructure and support from IT

- Easy for users to create, manage, and customize
- Project-centered or activity-oriented
- Strategic tool at the platform level; tactical tool at the team level
- Designed to facilitate enhanced collaboration
- Emphasis on user productivity and getting the job done

All of these teamware attributes can be applied to QuickPlace. IDC predicts a worldwide market opportunity for teamware solutions by the end of 2002, so there is ample opportunity to supply customized solutions for this market.

The best way to get acquainted with QuickPlace is to try it out. You can create your own test QuickPlace at the following Web address:
`http://www.quickplace.com`

11.2.1.1 QuickPlace features

Some of the things QuickPlace provides out of the box are:

Instant creation
A secure Web workspace can instantly be created for your team. The startup time is as quick as 60 seconds.

Shared content
Your team's discussions and documents are all maintained in a single location. Rich content can be created within or imported from Microsoft Office and other applications. Multiple revisions are easily tracked.

Shared sense of time
Gantt charts and calendars reflecting assigned tasks and meetings are available to help move the team toward its ultimate goals. Newsletters are e-mailed daily, providing an active pulse for the team. Real-time chat facilities are provided for instant contact.

Shared process and identity
As your project develops, the team develops a structure and process to meet its goals. The team's QuickPlace workspace easily adapts: *inner rooms* are available for sub-team security; browser-designed custom forms with approval cycles can be easily added; and the cosmetics of the QuickPlace can be quickly changed to reflect the team's emerging identity.

Shared knowledge
When the project or initiative is over, a new one is just around the corner. QuickPlace allows you to capture all the previously applied structure and knowledge gained into a PlaceType solution module. PlaceTypes can be

used as the basis for the next project/initiative by the same team or by other teams within your organization.

Deep customization
QuickPlace is designed to be easy to use and productive immediately out of the box. But for the corporate developer, Lotus Business Partner or Application Service Provider, QuickPlace provides a wealth of customization opportunities, including browser customization, HTML design, Java agents (PlaceBots) and server extensions. The gradient of customization extends QuickPlace from an “instant collaboration application” to an “instant collaboration platform.”

Directory integration
Domino has excellent directory support. In addition to the native Domino Name and Address book (now called Domino Directory), it supports LDAP directories (MS Exchange, NT Domain) and includes an automatic IIS detection and install option.

Microsoft Office 2000 integration
This feature gives your team the ability to author content from any Office application, and import templates as customized forms.

Task management
Quickplace lets you assign action items, track status, and monitor progress.

On-line awareness and chat
You can brainstorm new ideas or review content with on-line team members.

Themes and PlaceType solution modules
You can customize a QuickPlace workspace to mirror an existing intranet or corporate Web site, and create specific solution modules to address critical business issues that require immediate response and execution.

This is only a brief overview of what Lotus QuickPlace can do. For more information on what QuickPlace is and how to install and customize it, refer to *Customizing QuickPlace*, SG24-6000-00.

11.2.2 Installing QuickPlace on S/390

For the purpose of our portal, we used QuickPlace 2.0.5 beta code for Domino for S/390 R5.0.5.

Note

This is a very early beta release of QuickPlace on S/390. The install script was not available at the time of the beta, so we installed the product manually. We were not able to document the automated installation procedure; however, installation for the S/390 is expected to be similar to that for the Solaris.

11.2.2.1 Software levels

- QuickPlace 2.0.5
- Domino R5.05 (The release of QuickPlace must exactly match the release of Domino. This R2.0.5 code ONLY works on Domino R5.05.)

11.2.2.2 Restrictions

The following functions are not supported in this beta release:

- Offline: Use of this function will cause unpredictable failures, including possibly a server crash.
- Install: The QuickPlace install scripts have not been completed yet. Thus the installation instructions describe a manual process for installing the QuickPlace code on top of an existing Domino server. The install script provided here is an attempt by the developer to compromise between the amount of time he had to write it and the ease of using it. We should have the full install shortly and this process will go away.
- Stand-alone QuickPlace server: This beta release will only work when installed on top of an existing Domino server.

11.2.3 Integrating QuickPlace into the portal database

Once QuickPlace is set up, you will want to make it available to your users from the Portal Outline menu. Before you can do this, you need to log in to QuickPlace as the Administrator and register your QuickPlace server.

11.2.3.1 Entering QuickPlace for the first time

Start the Domino server. QuickPlace will be started when the HTTP server task is started. To enter QuickPlace, launch the following URL from a browser:

```
http://servername/quickplace
```

where servername is the IP address or hostname of the server on which you installed QuickPlace.

When the QuickPlace server is installed, the “Welcome” region of the “Administrator’s Place” is pre-configured to allow an entry point to the QuickPlace server. You as an administrator can then administer the newly installed QuickPlace server from this entry point. After signing in, you can begin to manage the QuickPlace server.

To get to the administration panel, go to the QuickPlace server’s main screen (also called the Welcome screen) and click **Sign In** in the top right-hand corner (refer to Figure 126).

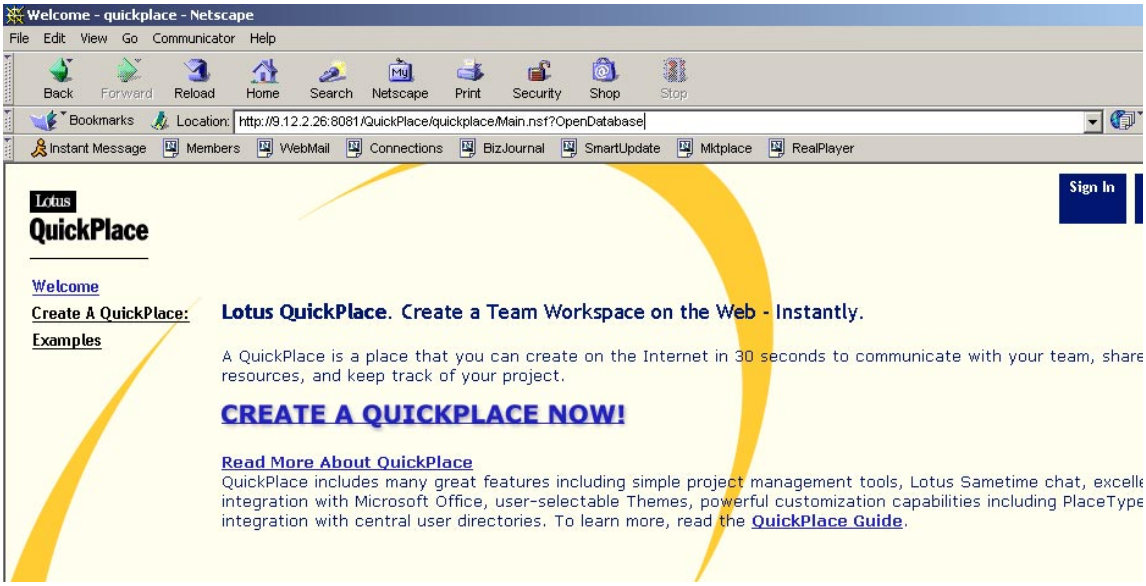

Welcome - quickplace - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Shop Stop
Bookmarks Location: http://9.12.2.26:8081/QuickPlace/quickplace/Main.nsf?OpenDatabase
Instant Message Members WebMail Connections BizJournal SmartUpdate Mktplace RealPlayer
Lotus
QuickPlace
Sign In
Welcome
Create A QuickPlace:
Examples
Lotus QuickPlace. Create a Team Workspace on the Web - Instantly.
A QuickPlace is a place that you can create on the Internet in 30 seconds to communicate with your team, share resources, and keep track of your project.
CREATE A QUICKPLACE NOW!
Read More About QuickPlace
QuickPlace includes many great features including simple project management tools, Lotus Sametime chat, excelle integration with Microsoft Office, user-selectable Themes, powerful customization capabilities including PlaceType integration with central user directories. To learn more, read the QuickPlace Guide.

Figure 126. Opening QuickPlace

Enter the user ID and the password of the administrator (this is the ID and password that was used during installation of the QuickPlace server) and click **Submit** (refer to Figure 127 on page 232).

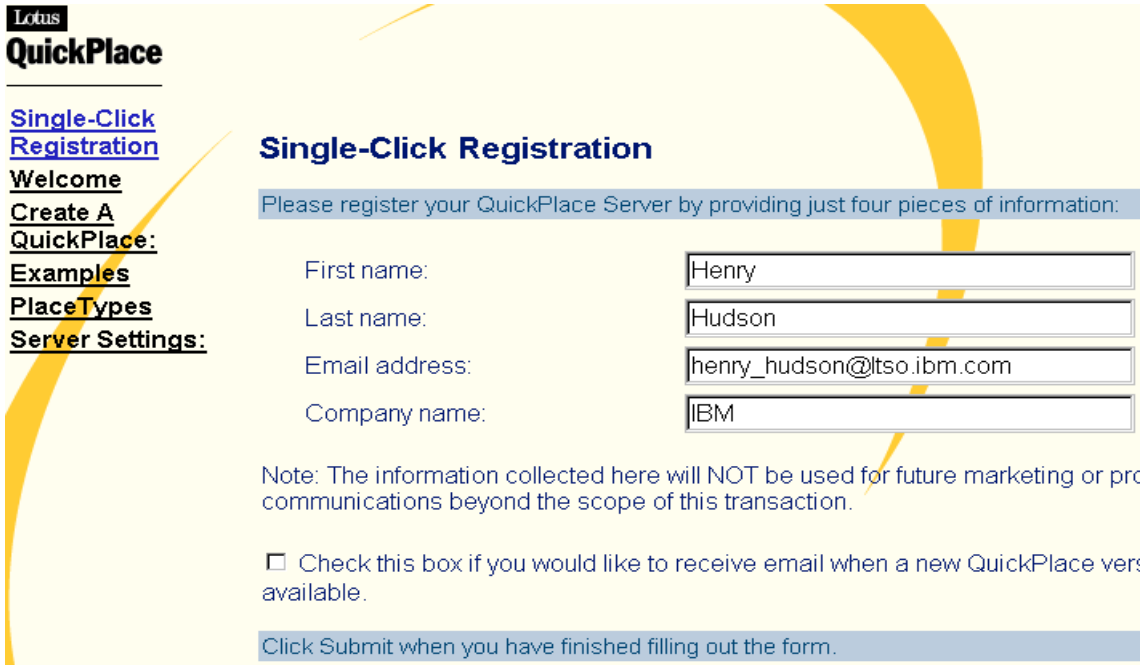


Figure 127. Registering your QuickPlace server

Upon successful sign-in, you see the Server Settings option in the left-hand side of the screen. Click **Server Settings** to go to the server administration screen. Each option is described there (Figure 128 on page 233). If necessary, click **Help** for more information.

Lotus
QuickPlace
[Go Back]
Instructions
Security
Mail
Existing QuickPlaces
User Directory
Other Options
Instructions
Server Settings. Welcome to the Server Settings room. O
administrators can enter. In the sidebar on your left are lin
areas:
Security
Enables you to control who can create QuickPlaces o
who can administer the server. It also allows you to
security (SSL).
Existing QuickPlaces
Displays a list of all the QuickPlaces on this server. Y
available disk space on your QuickPlace Server and c
which are no longer in use.
User Directory
Allows you to specify a central directory from which
can invite members.
Mail
Provides options that determine how this QuickPlace
receives email in a variety of network configurations.

Figure 128. QuickPlace server settings

11.2.3.2 Registering users

It is a good idea to register users who are allowed to use QuickPlace. This is more secure and allows you control over who can use the server and for what purposes.

To register users, click **User Directory**. This will take you to the User Directory screen, where you can select a directory to use for authentication (see Figure 129 on page 234).

You have three options to choose from:

- Microsoft Windows NT Domain

  If your network includes Windows NT Servers, this option allows you to select an NT Domain on the network. Members added from an NT Domain will be signed in using their current NT password.

- LDAP server

  LDAP is a standard way for servers on the Internet to present directory information. This choice allows you to add any LDAP server, including Lotus Domino and Microsoft Exchange servers. **(Note**: Your organization's network configuration may prohibit LDAP searches across the Internet.)

- Lotus Domino server

  If your network includes Lotus Domino servers, this option allows you to select a Domino server on the network. Members can be added from the available Domino directory and will be signed in using their current Internet password. **Note:** For lookup to work, this server must either be certified or cross-certified with the same cert.id as the directory server.

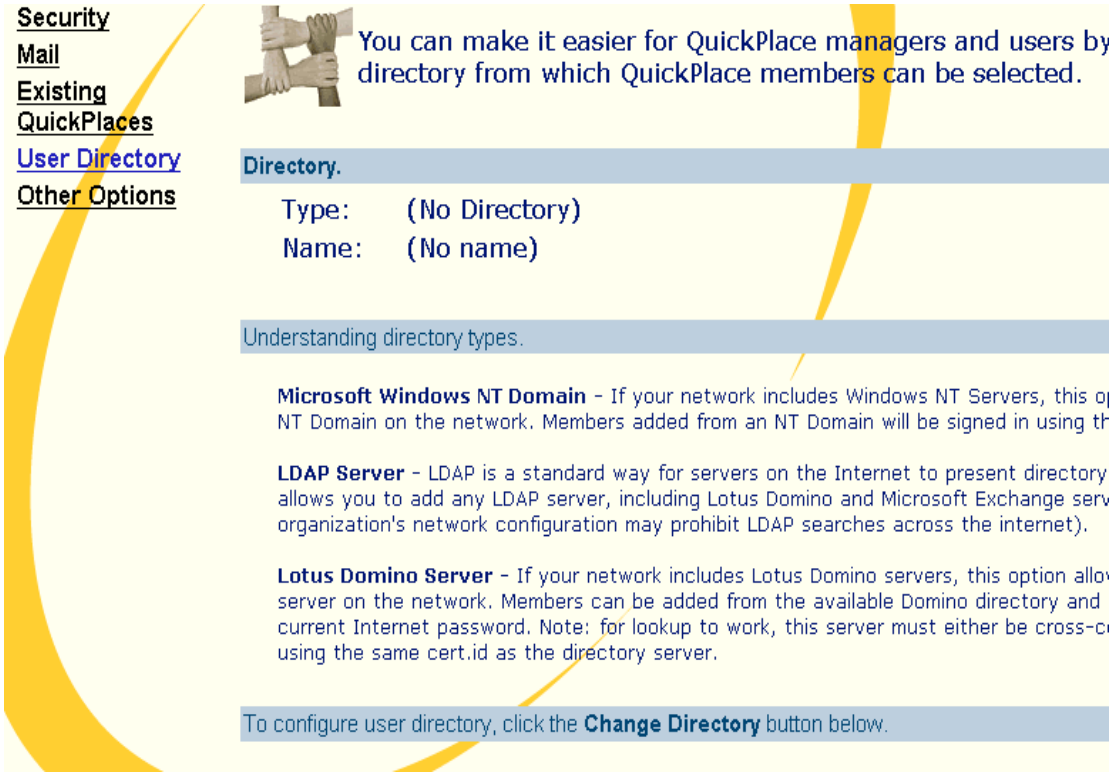

Security
Mail
Existing QuickPlaces
User Directory
Other Options
You can make it easier for QuickPlace managers and users by
directory from which QuickPlace members can be selected.
Directory.
Type: (No Directory)
Name: (No name)
Understanding directory types.
Microsoft Windows NT Domain - If your network includes Windows NT Servers, this o
NT Domain on the network. Members added from an NT Domain will be signed in using th
LDAP Server - LDAP is a standard way for servers on the Internet to present directory
allows you to add any LDAP server, including Lotus Domino and Microsoft Exchange serv
organization's network configuration may prohibit LDAP searches across the internet).
Lotus Domino Server - If your network includes Lotus Domino servers, this option allo
server on the network. Members can be added from the available Domino directory and
current Internet password. Note: for lookup to work, this server must either be cross-c
using the same cert.id as the directory server.
To configure user directory, click the Change Directory button below.

Figure 129. Registering users to use QuickPlace

When you click **Change Directory**, you will access a screen where you can specify the directory type and name (see Figure 130 on page 235). You can also specify whether you want managers of QuickPlaces to add new users or only to select them from the defined user directory.

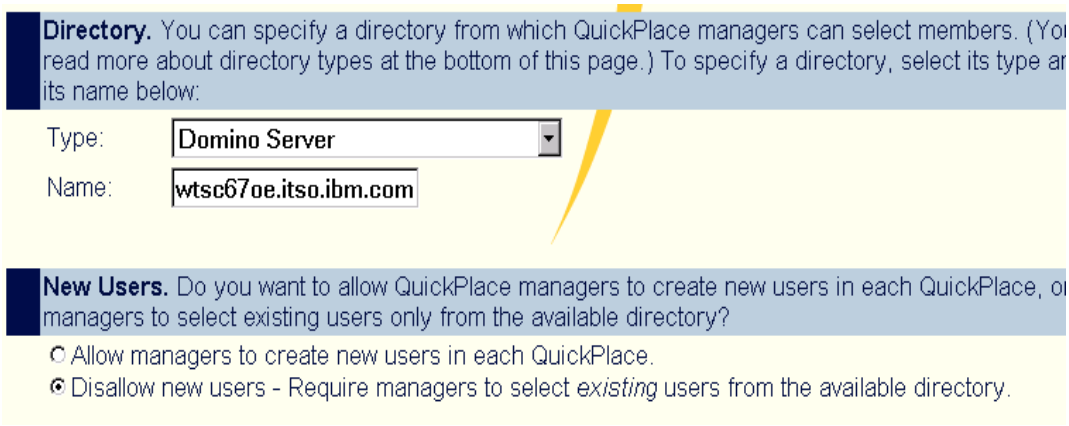

Directory. You can specify a directory from which QuickPlace managers can select members. (Yo
read more about directory types at the bottom of this page.) To specify a directory, select its type ar
its name below:
Type:
Domino Server
Name:
wtsc67oe.itso.ibm.com
New Users. Do you want to allow QuickPlace managers to create new users in each QuickPlace, o
managers to select existing users only from the available directory?
Allow managers to create new users in each QuickPlace.
Disallow new users - Require managers to select existing users from the available directory.

Figure 130. Specify a user directory for authentication

After you specified the directory from which users will be authenticated and set all the necessary server settings (Figure 128) according to your specific company standards, you can allow users to start using QuickPlace.

11.2.3.3 Creating a link from the portal database

Create the link to QuickPlace in the Outline of the portal database by adding an entry in the PortalMainMenu (see Figure 131 on page 236). Users can now start creating their own QuickPlaces and invite other users into the QuickPlace. Note that QuickPlace uses Domino HTTP Security (person documents in Domino Directories or LDAP directories) to log in users.

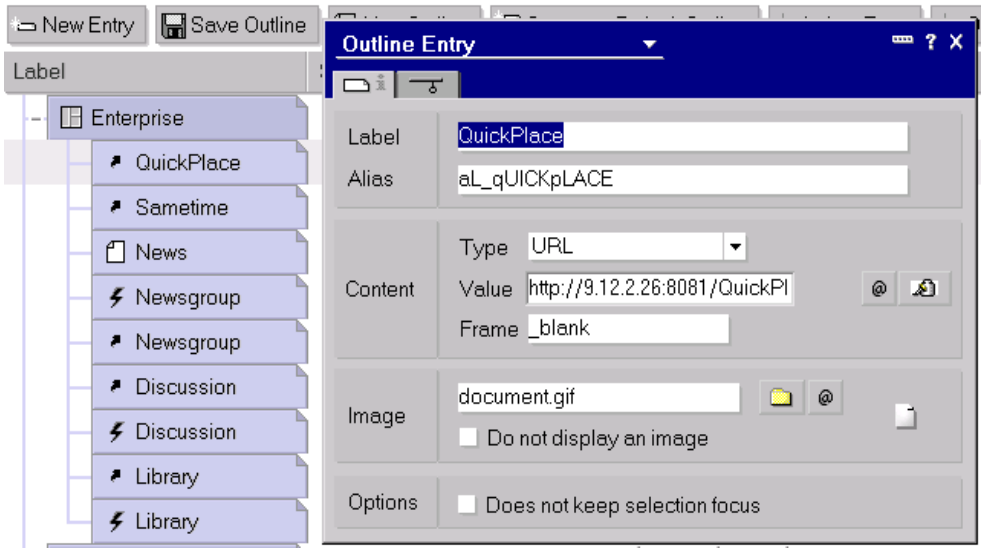


Figure 131. Creating a link to QuickPlace from the portal database

More information on setting up, customizing, and using QuickPlace is provided in the *QuickPlace User Manual* (at `www.notes.net/doc`) and *Customizing QuickPlace,* SG24-6000-00.

Appendix A. Code for the portal applications

Here is the source code for many of the infrastructure agents and utilities in our portal.

A.1 Invoices application (LCLSX)

For each agent, insert the `Uselsx "*LSXLC"` statement in the (Options) part of the agent code, so the LCLSX connector code gets loaded.

A.1.1 Update/insert invoice agent for Notes client access

```
Sub Initialize
    On Error Goto Handler
    ' Declare Notes Objects
    Dim NSession As New NotesSession
    Dim DB As NotesDatabase
    Dim Agent As NotesAgent
    Dim count As Long
    Dim ParmDoc As NotesDocument       'Parameter Doc = Employee Document

    'Declare LC Objects
    Dim Session As New LCSession
    Dim src As LCConnection
    Dim FieldLst As New LCFieldList
    Dim kfld As LCField
    Dim fld As LCField

    Set DB = NSession.currentdatabase
    Set Agent = NSession.currentagent

    'Read key Parameter from Employee Document
    Set ParmDoc = db.GetDocumentbyID(agent.ParameterDocID)
    Print "Parm Document opened, NoteID = "+ ParmDoc.NoteID
    tkey = Parmdoc.GetItemValue("INVNO" )
    Print "key " + tkey(0) + "fetched"

    'Establish DB2 Connection
    Session.ClearStatus

    Set src = New LCConnection ("db2") ''‘Create Connection object

    src.Database = "DB2G"
    src.Metadata = "DSN8610.INVOICES"
    src.UserID ="EBBERS1"
    src.Password = "EBBERS1"
    src.MapByName = True
    Print "Establishing Connection"

    src.Connect ''‘Connect to db2 data source ‘db2g’
    Print "Connection established"

    'Create key fieldlist to be updated or inserted
    Set kfld = FieldLst.append("INVNO", LCTYPE_TEXT)
    kfld.Flags = LCFIELDF_KEY
    kfld.value = tkey(0)
    Set fld = FieldLst.append("COMPNAME", LCTYPE_TEXT)
```

© Copyright IBM Corp. 2001

```
    fld.Text = Parmdoc.CustomerName(0)
    Set fld = FieldLst.append("CUSTNO", LCTYPE_TEXT)
    fld.Text = Parmdoc.CustomerNumber(0)
    Set fld = FieldLst.append("DATE", LCTYPE_TEXT)
    fld.Text = Parmdoc.Date(0)
    Set fld = FieldLst.append("ITEMS", LCTYPE_TEXT)
    fld.Text = Parmdoc.pitems(0)
    Set fld = FieldLst.append("SUM", LCTYPE_NUMERIC)
    fld.Value = Parmdoc.Sum(0)
    Set fld = FieldLst.append("PAID", LCTYPE_NUMERIC)
    fld.Value = Parmdoc.Paid(0)
    Set fld = FieldLst.append("STATUS", LCTYPE_TEXT)
    fld.Text = Parmdoc.Status(0)
    Set fld = FieldLst.append("COMMENT", LCTYPE_TEXT)
    fld.Text = Parmdoc.Comment(0)

    'Do a Update on the table if the row exists (key = EMPNO)
    count = src.Update(FieldLst,1,1)
    Print Cstr(count) + " rows updated"

    ' Do a Insert if row doesn't exist (key = EMPNO)
    If count = 0 Then
        count = src.Insert(FieldLst,1,1)
        Print Cstr(count) + " rows inserted"
    End If
    If count > 0 Then
        ret = ParmDoc.Remove(True)    'remove Document only if Table has been
updated/inserted
    End If
    src.Disconnect ''‘Disconnect from db2 data source ‘DB2G’
    Exit Sub
Handler:
    ''‘Error Handler
    If ( session.status <> LCSUCCESS) Then
        Print session.GetstatusText, 0, _
        + "The following Lotus Connector error has occurred."
    Else
        Print Error$, 0, _
        + "The following LotusScript error has occurred."
    End If
End Sub
```

A.1.2 Update/insert invoice agent for Web browser access

```
Sub Initialize
    On Error Goto Handler
    ' Declare Notes Objects
    Dim NSession As New NotesSession
    Dim DB As NotesDatabase
    Dim Agent As NotesAgent
    Dim count As Long
    Dim SQLString As String
    Dim hr_id As String
    Dim EmpDoc As NotesDocument'Fetched Row from DB2 Table
    Dim ParmDoc As NotesDocument       'Parameter Doc from Agent created by form in web
browser

    'Declare LC Objects
    Dim Session As New LCSession
    Dim src As LCConnection
    Dim FieldOut As New LCFieldList
```

```
Dim KeyList As New LCFieldList
Dim FLDInvNo As LCField
Dim FLDCompName As LCField
Dim FLDCustNo As LCField
Dim FLDDate As LCField
Dim FLDItems As LCField
Dim FLDSum As LCField
Dim FLDPaid As LCField
Dim FLDStatus As LCField
Dim FLDComment As LCField
Dim kfld As LCField

'Get Notes Docs
Set DB = NSession.currentdatabase

'Read key Parameter from Agent Parameter Document from Document Context (Web)
Set ParmDoc = Nsession.DocumentContext
tkey = Parmdoc.GetItemValue("InvoiceNo" )
Print "key " + tkey(0) + "fetched"

'Establish DB2 Connection
Session.ClearStatus

Set src = New LCConnection ("db2") ''‘Create Connection object

src.Database = "DB2G"
src.Metadata = "DSN8610.INVOICES"
src.UserID ="EBBERS1"
src.Password = "EBBERS1"
Print "Establishing Connection"

src.Connect ''‘Connect to db2 data source ‘DB2G’
Print "Connection established"

'Create key fieldlist
Set kfld = keylist.append("INVNO", LCTYPE_TEXT)
kfld.Flags = LCFIELDF_KEY

kfld.value = tkey(0)
Print "Key is "+ tkey(0)

'Do a keyed Select on the table
count = src.Select(keylist,1,fieldout)

'Set LC Fields
Set FLDInvNo = Fieldout.Lookup ("INVNO")
Set FLDCompname = Fieldout.Lookup ("COMPNAME")
Set FLDCustNo = Fieldout.Lookup ("CUSTNO")
Set FLDDate = Fieldout.Lookup ("DATE")
Set FLDItems = Fieldout.Lookup ("ITEMS")
Set FLDSum = Fieldout.Lookup ("SUM")
Set FLDPaid = Fieldout.Lookup ("PAID")
Set FLDStatus = Fieldout.Lookup ("STATUS")
Set FLDComment = Fieldout.Lookup ("COMMENT")

'Loop over Resultset
While (src.Fetch (Fieldout) > 0)

    'Create Document for each Row / Resultset entry
    Set empdoc = db.createdocument
    empdoc.form= "Invoice"
    empdoc.InvNo = FldInvno.text(0)
    empdoc.CustomerName = FldCompname.text(0)
```

```
        empdoc.Customernumber = FLDCustno.text(0)
        empdoc.Date = FLDDate.text(0)
        empdoc.PItems = FLDItems.text(0)
        empdoc.Sum = FLDSum.text(0)
        empdoc.Paid = FLDPaid.text(0)
        empdoc.Status = FLDStatus.text(0)
        empdoc.Comment = FLDComment.text(0)
        Call empdoc.Save (True, False)
        count = count + 1
    Wend

    src.Disconnect ''Disconnect from db2 data source 'DB2G'
    Exit Sub
Handler:
    ''Error Handler
    If ( session.status <> LCSUCCESS) Then
        Print session.GetstatusText, 0, _
        + "The following Lotus Connector error has occurred."
    Else
        Print Error$, 0, _
        + "The following LotusScript error has occurred."
    End If
End Sub
```

A.1.3 Delete invoice agent

```
Sub Initialize
    On Error Goto Handler
    ' Declare Notes Objects
    Dim NSession As New NotesSession
    Dim DB As NotesDatabase
    Dim Agent As NotesAgent
    Dim count As Long
    Dim ParmDoc As NotesDocument       'Parameter Doc = Employee Document

    'Declare LC Objects
    Dim Session As New LCSession
    Dim src As LCConnection
    Dim FieldLst As New LCFieldList
    Dim kfld As LCField
    Dim fld As LCField

    Set DB = NSession.currentdatabase
    Set Agent = NSession.currentagent

    'Read key Parameter from Employee Document
    Set ParmDoc = db.GetDocumentbyID(agent.ParameterDocID)
    Print "Parm Document opened, NoteID = "+ ParmDoc.NoteID
    tkey = Parmdoc.GetItemValue("INVNO" )
    Print "key " + tkey(0) + "fetched"

    'Establish DB2 Connection
    Session.ClearStatus

    Set src = New LCConnection ("db2") ''Create Connection object

    src.Database = "DB2G"
    src.Metadata = "DSN8610.INVOICES"
    src.UserID ="EBBERS1"
    src.Password = "EBBERS1"
    src.MapByName = True
    Print "Establishing Connection"
```

```
    src.Connect ''Connect to db2 data source
    Print "Connection established"

    'Create key fieldlist with key of the row to be removed
    Set kfld = FieldLst.append("INVNO", LCTYPE_TEXT)
    kfld.Flags = LCFIELDF_KEY
    kfld.value = tkey(0)

    'Do a Delete on the table if the row exists (key = INVNO)
    count = src.Remove(FieldLst,1,1)
    Print Cstr(count) + " rows deleted"

    If count > 0 Then
        ret = ParmDoc.Remove(True)    'remove Document only if row has been deleted
        Print "deleted"
    End If
    src.Disconnect 'Disconnect from db2 data source 'DB2G'
    Exit Sub

Handler:
    ''Error Handler
    If ( session.status <> LCSUCCESS) Then
        Print session.GetstatusText, 0, _
        + "The following Lotus Connector error has occurred."
    Else
        Print Error$, 0, _
        + "The following LotusScript error has occurred."
    End If
End Sub
```

A.2 Parts ordering application (MQLSX)

For each agent, this statement was added to the Options section of the agent:

```
Uselsx "mqlsx"
```

A.2.1 The WebMQPutOrder agent

```
Sub Initialize
    'Declare MQObjects
    Dim mqs         As MQSession
    Dim mqqm        As MQQueueManager
    Dim mqq         As MQQueue
    Dim mqm         As MQMessage
    Dim mqpmo       As MQPutMessageOptions
    Dim mqgmo       As MQGetMessageOptions
    Dim mqp         As MQProcess

    Dim QueueName   As String

    Dim T           As String
    Dim ErrString   As String
    Dim PutMsg      As Integer
    Dim GetMsg      As Integer
    Dim msgout      As String
    Dim msgin       As String
```

```
    Dim db As NotesDatabase
    Dim session As NotesSession
    Dim Agent As NotesAgent
    Dim ParmDoc As NotesDocument

    Set session = New NotesSession
    Set db = session.CurrentDatabase

    'Read MQ string values from DocumentContext (Document where agent is activated)
    Set ParmDoc = session.DocumentContext
    Stat$= ParmDoc.Status(0)
    CustNumber$ = ParmDoc.CustomerNumber(0)
    CustName$ = ParmDoc.CustomerName(0)
    Item$ = ParmDoc.Item(0)
    Phone$ = Cstr(ParmDoc.NoteID) & "     "
    OrderDate$ = Cstr(ParmDoc.Date(0))
    Amount$ = Cstr(ParmDoc.Amount(0))
    Comment$ = ParmDoc.Comment(0)

    'Specifiy name of the Queue to put the message in
    QueueName$ = "SG245682.NOTES.OUT.QUEUE"

    On Error Goto Err_Routine
    On Error 32000  Resume Next

' Creating New MQSession
    Set mqs = New MQSession

    If mqs.CompletionCode <> MQCC_OK Then
        Print "Error: MQSession cc: " & mqs.CompletionCode
        Print "Error: MQSession rc: " & mqs.ReasonCode
    End If

' Initiate MQQueueManager, MQPutMessageOptions, Message
    Set mqqm = New MQQueueManager
    Set mqpmo = New MQPutMessageOptions
    Set mqm = New MQMessage

' mqqm.Connect
    QmgrNames$ = "MQSX"
    mqqm.name = QmgrNames$

    Call mqqm.Connect

' Access the queue
    QOpenOptions& = 0

    Set mqq = mqqm.AccessQueue(QueueName$, QOpenOptions&, "","","")

' PUT Msg
'---------------
    ' Write string into Queue
    mqm.WriteString(Stat$ & CustNumber$ & CustName$ & Item$ & Phone$ & OrderDate$ &
Amount$ & Comment$)

    If mqm.CompletionCode <> MQCC_OK Then
        Print "Error: MQSession cc: " & mqs.CompletionCode
        Print "Error: MQSession rc: " & mqs.ReasonCode
    Else
        Print "Your Order is shipped.   You will receive the confirmation shortly."
    End If

        'setting MessageType to MQMT_DATAGRAM
```

```
    mqm.Messagetype = MQMT_DATAGRAM
    mqm.Format = MQFMT_STRING

     ' Put the msg
    Call mqq.put(mqm,mqpmo)

     ' Commit the message
    Call mqqm.Commit

' Disconnect
    Call mqqm.Disconnect

    Goto End_Routine


Err_Routine:
    ErrString = "Err_Routine: Error " & Err() & " on line " & Erl() & "(" & Error$(Err())
& ")"
    Print ErrString

    If Not ( mqp   Is Nothing ) Then
        Call mqp.ClearErrorCodes
    End If

    If Not ( mqpmo Is Nothing ) Then
        Call mqpmo.ClearErrorCodes
    End If

    If Not ( mqgmo Is Nothing ) Then
        Call mqgmo.ClearErrorCodes
    End If

    If Not ( mqm   Is Nothing ) Then
        Call mqm.ClearErrorCodes
    End If

    If Not ( mqq   Is Nothing ) Then
        Call mqq.ClearErrorCodes
    End If

    If Not ( mqqm  Is Nothing ) Then
        Call mqqm.ClearErrorCodes
    End If

    If Not ( mqs   Is Nothing ) Then
        Call mqs.ClearErrorCodes
    End If

    Resume Next

'-----------------------------------------------------------------------
End_Routine:

    If Not ( mqp   Is Nothing ) Then
        Call mqp.ClearErrorCodes
        Delete mqp
    End If

    If Not ( mqpmo Is Nothing ) Then
        Call mqpmo.ClearErrorCodes
        Delete mqpmo
    End If
```

```
    If Not ( mqgmo Is Nothing ) Then
        Call mqgmo.ClearErrorCodes
        Delete mqgmo
    End If

    If Not ( mqm   Is Nothing ) Then
        Call mqm.ClearErrorCodes
        Delete mqm
    End If

    If Not ( mqq   Is Nothing ) Then
        Call mqq.ClearErrorCodes
        Delete mqq
    End If

    If Not ( mqqm  Is Nothing ) Then
        Call mqqm.ClearErrorCodes
        Delete mqqm
    End If

    If Not ( mqs   Is Nothing ) Then
        Call mqs.ClearErrorCodes
        Delete mqs
    End If


End Sub
```

A.2.2 The MQGetOrderStatus agent

```
Sub Initialize
    'Declare MQ Objects
    Dim mqs        As MQSession
    Dim mqqm       As MQQueueManager
    Dim mqq        As MQQueue
    Dim mqm        As MQMessage
    Dim mqpmo      As MQPutMessageOptions
    Dim mqgmo      As MQGetMessageOptions
    Dim mqp        As MQProcess

    Dim QueueName  As String

    Dim T          As String
    Dim ErrString  As String
    Dim PutMsg     As Integer
    Dim GetMsg     As Integer
    Dim msgout     As String
    Dim msgin      As String

    Dim db As NotesDatabase
    Dim session As NotesSession
    Dim Agent As NotesAgent
    Dim ParmDoc As NotesDocument

    Set session = New NotesSession
    Set db = session.CurrentDatabase
    Set Agent = session.currentagent

    Print "Agent QMGetOrderStatus started"

    'Specify Name of Queue to poll
    QueueName$ = "SG245682.NOTES.IN.QUEUE"
```

```
    On Error Goto Err_Routine
    On Error 32000  Resume Next

' Creating New MQSession
    Set mqs = New MQSession

    If mqs.CompletionCode <> MQCC_OK Then
        Print "Error: MQSession cc: " & mqs.CompletionCode
        Print "Error: MQSession rc: " & mqs.ReasonCode
    End If

' Initiate MQQueueManager, MQPutMessageOptions, Message
    Set mqqm = New MQQueueManager
    Set mqgmo = New MQGetMessageOptions
    Set mqm = New MQMessage

' mqqm.Connect to Queue Manager MQSX
    QmgrNames$ = "MQSX"
    mqqm.name = QmgrNames$

    Call mqqm.Connect

' Access the queue
    QOpenOptions& = 0
    Set mqq = mqqm.AccessQueue(QueueName$, QOpenOptions&, "","","")


' GET MSG
'---------------
        'Setting Message Options
    mqgmo.options = MQGMO_FAIL_IF_QUIESCING + _
    MQGMO_SYNCPOINT

    T$ = "17.Queue depth before get: " & mqq.CurrentDepth
    Print T$                                        'Check # Queue entries

     ' Get the message
    Call mqq.Get(mqm,mqgmo)

    If mqs.CompletionCode <> MQCC_OK Then
        Print "Error: " & T$ & " cc: " & mqs.CompletionCode
        Print "Error: " & T$ & " rc: " & mqs.ReasonCode
    End If

    Print "Got the Message"

     ' Read the message
    msgin$ = mqm.ReadString(mqm.MessageLength)

    Print "Read the Message"

     ' Commit the get
    Call mqqm.Commit

    T$ = "24.After Get and COMMIT queue depth is: " & mqq.CurrentDepth
    Print T$                                    'Check # Queue entries

' Disconnect
    Print "1"
    Call mqqm.Disconnect
    Print "2"
'Read Data into form of Document searched with NoteID
```

```
    DocID$ = Trim$(Mid$(msgin$,48,8))             'Get NoteID of corresponding Doc from
Queue string
    Set ParmDoc = db.GetDocumentbyID(DocID$)
    Print "Document opened"
    ParmDoc.Status = Left$(msgin$,1)
    ParmDoc.CustomreNumber = Mid$(msgin$,2,6)
    ParmDoc.CustomerName = Mid$(msgin$,8,20)
    ParmDoc.Item = Mid$(msgin$,28,20)
    ParmDoc.Date = Mid$(msgin$,56,8)
    ParmDoc.Amount = Mid$(msgin$,64,8)
    ParmDoc.Comment = Mid$(msgin$,72,9)
    Call ParmDoc.Save(True, False)

    Goto End_Routine


Err_Routine:
    ErrString = "Err_Routine: Error " & Err() & " on line " & Erl() & "(" & Error$(Err())
& ")"
    Print ErrString

    If Not ( mqp   Is Nothing ) Then
       Call mqp.ClearErrorCodes
    End If

    If Not ( mqpmo Is Nothing ) Then
       Call mqpmo.ClearErrorCodes
    End If

    If Not ( mqgmo Is Nothing ) Then
       Call mqgmo.ClearErrorCodes
    End If

    If Not ( mqm   Is Nothing ) Then
       Call mqm.ClearErrorCodes
    End If

    If Not ( mqq   Is Nothing ) Then
       Call mqq.ClearErrorCodes
    End If

    If Not ( mqqm  Is Nothing ) Then
       Call mqqm.ClearErrorCodes
    End If

    If Not ( mqs   Is Nothing ) Then
       Call mqs.ClearErrorCodes
    End If

    Resume Next

'---------------------------------------------------------------------------
End_Routine:

    If Not ( mqp   Is Nothing ) Then
       Call mqp.ClearErrorCodes
       Delete mqp
    End If

    If Not ( mqpmo Is Nothing ) Then
       Call mqpmo.ClearErrorCodes
       Delete mqpmo
    End If
```

```
    If Not ( mqgmo Is Nothing ) Then
        Call mqgmo.ClearErrorCodes
        Delete mqgmo
    End If

    If Not ( mqm   Is Nothing ) Then
        Call mqm.ClearErrorCodes
        Delete mqm
    End If

    If Not ( mqq   Is Nothing ) Then
        Call mqq.ClearErrorCodes
        Delete mqq
    End If

    If Not ( mqqm  Is Nothing ) Then
        Call mqqm.ClearErrorCodes
        Delete mqqm
    End If

    If Not ( mqs   Is Nothing ) Then
        Call mqs.ClearErrorCodes
        Delete mqs
    End If


End Sub
```

A.3 The personal page generator

Here is an agent we used when creating the personal page of the portal.

A.3.1 The AgentGetMailUrl agent

```
Option Public
Use "DomDir"

Dim Session As New NotesSession
    Dim Db As NotesDatabase
    Dim DomDir As NotesDatabase
    Dim Doc As Notesdocument
    Dim PersDocColl As NotesDocumentcollection
    Dim PerDoc As NotesDocument
    Dim ServDocColl As NotesDocumentcollection
    Dim ServDoc As NotesDocument

    Dim agentLog As New NotesLog("Agent log")
    Call agentLog.OpenAgentLog

     'Get the full path of this database
    Call agentLog.LogAction("Get the full path of this database")
    Set Db = Session.CurrentDatabase
    Set DomDir = New Notesdatabase("",DomDirPath$)
    DbPath$ = Db.filepath
```

```
    Set doc = Session.DocumentContext
    NameUrl$ = Lcase$(doc.user(0))
    Call agentLog.LogAction("NameUrl$ = "+doc.user(0))
     'Search the mailbox via Persondocument entry
    SearchStr$ = "Form = ""Person"" & @lowercase(Fullname) = """ + NameUrl$ + """"

    Set PersDocColl = DomDir.Search(SearchStr$, Nothing, 0)

    'In case of multiple matches
    If PersDocColl.count > 1 Then   'multiple hierarchical name or rep. conflicts
        Call agentLog.LogAction("multiple hierarchical name or rep. conflicts")
        Print "<b>ERROR</b>The user has one or more conflicting names.<p>"
        Print "<b>Contact the Webmaster</b>"
        Print "<p><a href=" "OpenMail?OpenForm" " >Go back</a>"
        Exit Sub
    End If

    'Get Persondoc entries
    Call agentLog.LogAction("Get Persondoc entries")
    Set PersDoc = PersDocColl.GetFirstDocument
    Call agentLog.LogAction("Set PersDoc = PersDocColl.GetFirstDocument")
    If Not (PersDoc Is Nothing) Then

        'Replace \ with /
        Call agentLog.LogAction("Replace \ with /")
        Temp1$ = Trim(PersDoc.mailfile(0))
        pos = Instr(Temp1$,"\")
        Temp$ = Left$(Temp1$,pos-1) + "/" + Mid$(Temp1$,pos+1)

        If Instr(PersDoc.MailFile(0),".nsf") Then
            mail$ = Temp$
        Else
            mail$ = Temp$+ ".nsf"
        End If
    End If

    Server$ = PersDoc.MailServer(0)
    Call agentLog.LogAction("Server$ = PersDoc.MailServer(0)")

    SearchStr$ = "Form = ""Server"" & Servername = """ + Server$ + """"
    Set ServDocColl = DomDir.Search(SearchStr$, Nothing, 0)
    Set ServDoc = ServDocColl.GetFirstDocument
    If Not (ServDoc Is Nothing) Then
        Call agentLog.LogAction("For URL we use the NetAdr_0 field in the server
document")

        'For URL we use the NetAdr_0 field in the server document
        address$ = ServDoc.NetAddr_0(0)

        doc.Mail ="'http://"+address$ + "/" + mail$ + "/($Inbox)?OpenView'"
        doc.Calendar = "'http://"+address$ + "/" + mail$ + "/($Calendar)?OpenView'"
        doc.ToDo = "'http://"+address$ + "/" + mail$ + "/($ToDo)?OpenView'"
        doc.Subscription =
"'http://wtsc67oe/Applications/Subscrip.nsf/SubscriptionsFullscreen'"
        Call doc.save(True,True)
        Exit Sub
    Else
        Call agentLog.LogAction("<b>ERROR</b>, mailfile or server not found.")
        Print "<b>ERROR</b>, mailfile or server not found."
        Print "<p><a href=""/" + dbpath$ + """>Go back</a> "
    End If
    Call agentLog.Close
End Sub
```

A.4 The menu generator interface

Here are agents we used in the menu generator portion of the portal infrastructure.

A.4.1 The AddOutlineEntryWeb agent

```
Option Public

Sub Initialize
    Dim session As New NotesSession
    Dim db As NotesDatabase
    Dim doc As NotesDocument
    Dim doc2 As NotesDocument
    Dim outline As NotesOutline
    Dim NewEntry As NotesOutlineEntry

    Dim agentLog As New NotesLog("Agent log")
    Call agentLog.OpenAgentLog
    Call agentLog.LogAction("Begin AddOutlineEntryWeb")

    Set db = session.CurrentDatabase
    Set doc = Session.DocumentContext

    Call agentLog.LogAction("Before read parameters")
    'Get the parameters data
    FunctionalPage = doc.FunctionalPage(0)
    Call agentLog.LogAction(Cstr(FunctionalPage))
    URLChoose = doc.URLChoose(0)
    Call agentLog.LogAction(Cstr(URLChoose))
    TitleChoose = doc.TitleChoose(0)
    Call agentLog.LogAction(Cstr(TitleChoose))
    FuncPage = doc.FuncPage
    OutlineMenu = doc.OutlineMenu
    Call agentLog.LogAction("After read parameters")

    'Get the corresponding Outline Menu
    NumElements = Ubound(FuncPage)
    Call agentLog.LogAction("Elements:= " + Cstr(NumElements))
    For i = 0 To NumElements
        Call agentLog.LogAction("into the loop")
        If ( Cstr(FunctionalPage) = Cstr(FuncPage(i)) )Then
            Menu = OutlineMenu(i)
            Set outline = db.GetOutline(Cstr(Menu))
            FrameName = "Body"
            Call agentLog.LogAction(Cstr(Menu))
            Exit For
        End If
    Next

    Set NewEntry = outline.Createentry(Cstr(TitleChoose))
    Call agentLog.LogAction("Generates the new entry")

    If ( NewEntry.SetURL(Cstr(URLChoose)) ) Then
        Call agentLog.LogAction("URL assigned")
        NewEntry.FrameText = Cstr(FrameName)
        Call agentLog.LogAction("FrameSet assigned")
    Else
        Call agentLog.LogAction("URL not assigned")
    End If
```

```
    Call outline.save()

    Call agentLog.LogAction("Add OutlineEntry")

    'Generate the corresponding functional choose document for adminstration proposals
    Set doc2 = db.CreateDocument
    doc2.Form  = "FunctionalChooses"
    doc2.FunctionalPage = Cstr(FunctionalPage)
    doc2.ChooseName = Cstr(TitleChoose)
    Call doc2.Save( True, True )

    Call agentLog.LogAction("Add corresponding fuctional choose document")
    Call agentLog.LogAction("End AddOutlineEntryWeb")

    Call agentLog.Close

End Sub
```

A.4.2 The RemoveOutlineEntryWeb agent

```
Option Public

Sub Initialize
    Dim session As New NotesSession
    Dim db As NotesDatabase
    Dim doc As NotesDocument
    Dim doc2 As NotesDocument
    Dim outline As NotesOutline
    Dim entry As NotesOutlineEntry
    Dim View As NotesView

    Dim agentLog As New NotesLog("Agent log")
    Call agentLog.OpenAgentLog
    Call agentLog.LogAction("Begin RemoveOutlineEntryWeb")

    Set db = session.CurrentDatabase
    Set doc = Session.DocumentContext

    Call agentLog.LogAction("Before read parameters")
    'Get the parameters data
    FunctionalPage = doc.FunctionalPage(0)
    Call agentLog.LogAction(Cstr(FunctionalPage))
    Options = doc.Options(0)
    Call agentLog.LogAction(Cstr(Options))
    FuncPage = doc.FuncPage
    OutlineMenu = doc.OutlineMenu
    Call agentLog.LogAction("After read parameters")

    'Get the corresponding Outline Menu
    NumElements = Ubound(FuncPage)
    Call agentLog.LogAction("Elements:= " + Cstr(NumElements))
    For i = 0 To NumElements
        Call agentLog.LogAction("into the loop")
        If ( Cstr(FunctionalPage) = Cstr(FuncPage(i)) )Then
            Menu = OutlineMenu(i)
            Set outline = db.GetOutline(Cstr(Menu))
            Call agentLog.LogAction("Outline: " + Cstr(Menu))
            Exit For
        End If
    Next
```

```
    'Get the required entry
    Set entry = outline.GetFirst()

    If Not(entry Is Nothing) Then

        Do

            If ( Cstr(entry.Label) = Cstr(Options) ) Then

                Call outline.RemoveEntry(entry)
                Call agentLog.LogAction("OutlineEntry Removed")

                ' Get the functional choose doc
                Set view = db.GetView("ViewFuncChoose")
                Set doc2 = view.GetFirstDocument

                If Not(doc2 Is Nothing) Then

                    Do
                        If (  (Cstr(doc2.FunctionalPage(0)) = Cstr(FunctionalPage)) And
(Cstr(doc2.ChooseName(0)) = Cstr(Options)) )Then
                            Call doc2.Remove(True)
                            Call agentLog.LogAction("Fuctional choose removed")
                            Exit Do
                        End If
                        Set doc2 = view.GetNextDocument(doc2)

                    Loop While ( Not(doc2 Is Nothing) )

                End If

                Exit Do

            End If

            Set entry = outline.GetNext(entry)

        Loop While Not(entry Is Nothing)

    End If

    Call outline.save()

    Call agentLog.LogAction("End RemoveOutlineEntryWeb")
    Call agentLog.Close

End Sub
```

A.4.3 The UpdateOutlineEntryWeb agent

```
Option Public

Sub Initialize
    Dim session As New NotesSession
    Dim db As NotesDatabase
    Dim doc As NotesDocument
    Dim doc2 As NotesDocument
    Dim outline As NotesOutline
    Dim entry As NotesOutlineEntry
    Dim View As NotesView

    Dim agentLog As New NotesLog("Agent log")
```

```
    Call agentLog.OpenAgentLog
    Call agentLog.LogAction("Begin UpdateOutlineEntryWeb")

    Set db = session.CurrentDatabase
    Set doc = Session.DocumentContext

    Call agentLog.LogAction("Before read parameters")
    'Get the parameters data
    FunctionalPage = doc.FunctionalPage(0)
    Call agentLog.LogAction(Cstr(FunctionalPage))
    Options = doc.Options(0)
    Call agentLog.LogAction(Cstr(Options))
    TitleChoose = doc.TitleChoose(0)
    Call agentLog.LogAction(Cstr(TitleChoose))
    URLChoose = doc.URLChoose(0)
    Call agentLog.LogAction(Cstr(URLChoose))
    FuncPage = doc.FuncPage
    OutlineMenu = doc.OutlineMenu
    Call agentLog.LogAction("After read parameters")

    'Get the corresponding Outline Menu
    NumElements = Ubound(FuncPage)
    Call agentLog.LogAction("Elements:= " + Cstr(NumElements))
    For i = 0 To NumElements
        Call agentLog.LogAction("into the loop")
        If ( Cstr(FunctionalPage) = Cstr(FuncPage(i)) )Then
            Menu = OutlineMenu(i)
            Set outline = db.GetOutline(Cstr(Menu))
            Call agentLog.LogAction("Outline: " + Cstr(Menu))
            Exit For
        End If
    Next

    'Get the required entry
    Set entry = outline.GetFirst()

    If Not(entry Is Nothing) Then

        Do

            Call agentLog.LogAction("Compare: " + Cstr(entry.Label) + " / " +
Cstr(Options) )
            If ( Cstr(entry.Label) = Cstr(Options) ) Then

                'Set the label
                If ( Cstr(TitleChoose)  <> "") Then
                    entry.Label =  Cstr(TitleChoose)
                    Call agentLog.LogAction("Entry label changed")
                Else
                    Call agentLog.LogAction("Entry label not changed")
                End If

                'Set the new URL
                If  (Cstr(URLChoose) <> "") Then

                    Call agentLog.LogAction("To modify the URL")

                    If ( entry.SetURL(Cstr(URLChoose))) Then
                        Call agentLog.LogAction("URL modified")
                    Else
                        Call agentLog.LogAction("URL not assigned")
                    End If
```

```
                Else
                    Call agentLog.LogAction("URL not modified")
                End If

                ' Get the functional choose doc
                Set view = db.GetView("ViewFuncChoose")
                Set doc2 = view.GetFirstDocument

                Call agentLog.LogAction("looking for  the functional choose doc")
                If Not(doc2 Is Nothing) Then

                   Do
                       If (  (Cstr(doc2.FunctionalPage(0)) = Cstr(FunctionalPage)) And
(Cstr(doc2.ChooseName(0)) = Cstr(Options)) )Then

                          'Set the label
                          If ( Cstr(TitleChoose)  <> "") Then
                             doc2.ChooseName =  Cstr(TitleChoose)
                          End If

                          Call doc2.save(True, True)

                          Call agentLog.LogAction("Fuctional choose modified")
                          Exit Do

                       End If
                       Set doc2 = view.GetNextDocument(doc2)

                   Loop While ( Not(doc2 Is Nothing) )

                End If

                Exit Do

             End If

             Set entry = outline.GetNext(entry)

          Loop While Not(entry Is Nothing)

       End If

       Call outline.save()

       Call agentLog.LogAction("End UpdateOutlineEntryWeb")
       Call agentLog.Close

End Sub
```

A.4.4 The AddOutlineEntryNotes agent

```
Option Public

Sub Initialize

    Dim workspace As New NotesUIWorkspace
    Dim uidoc As NotesUIDocument
    Dim db As New NotesDatabase( "", "" )
    Dim doc As NotesDocument
    Dim doc2 As NotesDocument
    Dim outline As NotesOutline
    Dim NewEntry As NotesOutlineEntry
```

```
    Dim agentLog As New NotesLog("Agent log")
    Call agentLog.OpenAgentLog
    Call agentLog.LogAction("Begin AddOutlineEntryNotes")

    Call db.Open( "Wtsc67oe", "PortalS390.nsf" )
    Set uidoc = workspace.CurrentDocument
    Set doc = uidoc.Document

    'Get the parameters data
    Call agentLog.LogAction("Before read parameters")
    FunctionalPage = doc.FunctionalPage(0)
    URLChoose = doc.URLChoose(0)
    TitleChoose = doc.TitleChoose(0)
    FuncPage = doc.FuncPage
    OutlineMenu = doc.OutlineMenu
    Call agentLog.LogAction("After read parameters")

    'Get the corresponding Outline Menu
    NumElements = Ubound(FuncPage)
    Call agentLog.LogAction("Elements:= " + Cstr(NumElements))
    For i = 0 To NumElements
        Call agentLog.LogAction("into the loop")
        If ( Cstr(FunctionalPage) = Cstr(FuncPage(i)) )Then
            Menu = OutlineMenu(i)
            Set outline = db.GetOutline(Cstr(Menu))
            FrameName = "Body"
            Call agentLog.LogAction(Cstr(Menu))
            Exit For
        End If
    Next

    Set NewEntry = outline.Createentry(Cstr(TitleChoose))
    Call agentLog.LogAction("Genero el nuevo entry")

    If ( NewEntry.SetURL(Cstr(URLChoose)) Then
        Call agentLog.LogAction("URL assigned")
        NewEntry.FrameText = Cstr(FrameName)
        Call agentLog.LogAction("FrameSet assigned")
    Else
        Call agentLog.LogAction("URL not assigned")
    End If

    Call outline.save()

    Call agentLog.LogAction("Add OutlineEntry")

    'Generate the corresponding functional choose document for adminstration proposals
    Set doc2 = db.CreateDocument
    doc2.Form  = "FunctionalChooses"
    doc2.FunctionalPage = Cstr(FunctionalPage)
    doc2.ChooseName = Cstr(TitleChoose)
    Call doc2.Save( True, True )

    'Call db.UpdateFTIndex( False )
    'Call workspace.ReloadWindow( )

    Call agentLog.LogAction("Add corresponding fuctional choose document")
    Call agentLog.LogAction("End AddOutlineEntryNotes")

    Call agentLog.Close

End Sub
```

A.4.5 The RemoveOutlineEntryNotes agent

```
Option Public

Sub Initialize
    Dim workspace As New NotesUIWorkspace
    Dim uidoc As NotesUIDocument
    Dim db As New NotesDatabase( "", "" )
    Dim doc As NotesDocument
    Dim doc2 As NotesDocument
    Dim outline As NotesOutline
    Dim entry As NotesOutlineEntry
    Dim View As NotesView

    Dim agentLog As New NotesLog("Agent log")
    Call agentLog.OpenAgentLog
    Call agentLog.LogAction("Begin RemoveOutlineEntryNotes")

    Call db.Open( "Wtsc67oe", "PortalS390.nsf" )
    Set uidoc = workspace.CurrentDocument
    Set doc = uidoc.Document

    'Get the parameters data
    Call agentLog.LogAction("Before read parameters")
    FunctionalPage = doc.FunctionalPage(0)
    Options = doc.Options(0)
    FuncPage = doc.FuncPage
    OutlineMenu = doc.OutlineMenu
    Call agentLog.LogAction("After read parameters")

    'Get the corresponding Outline Menu
    NumElements = Ubound(FuncPage)
    Call agentLog.LogAction("Elements:= " + Cstr(NumElements))
    For i = 0 To NumElements
        If ( Cstr(FunctionalPage) = Cstr(FuncPage(i)) )Then
            Menu = OutlineMenu(i)
            Set outline = db.GetOutline(Cstr(Menu))
            Call agentLog.LogAction("Outline: " + Cstr(Menu))
            Exit For
        End If
    Next

    'Get the required entry
    Set entry = outline.GetFirst()

    If Not(entry Is Nothing) Then

        Do

            If ( Cstr(entry.Label) = Cstr(Options) ) Then

                Call outline.RemoveEntry(entry)
                Call agentLog.LogAction("OutlineEntry Removed")

                ' Get the functional choose doc
                Set view = db.GetView("ViewFuncChoose")
                Set doc2 = view.GetFirstDocument

                If Not(doc2 Is Nothing) Then

                    Do
                        If (  (Cstr(doc2.FunctionalPage(0)) = Cstr(FunctionalPage)) And
(Cstr(doc2.ChooseName(0)) = Cstr(Options)) )Then
```

```
                        Call doc2.Remove(True)
                        Call agentLog.LogAction("Fuctional choose removed")
                        Exit Do
                    End If
                    Set doc2 = view.GetNextDocument(doc2)

                Loop While ( Not(doc2 Is Nothing) )

            End If

            Exit Do

        End If

        Set entry = outline.GetNext(entry)

    Loop While Not(entry Is Nothing)

End If

Call outline.save()

'Call db.UpdateFTIndex( False )
'Call workspace.ReloadWindow( )

Call agentLog.LogAction("End RemoveOutlineEntryNotes")
Call agentLog.Close

End Sub
```

A.4.6 The UpdateOutlineEntryNotes agent

```
Option Public

Sub Initialize
    Dim workspace As New NotesUIWorkspace
    Dim uidoc As NotesUIDocument
    Dim db As New NotesDatabase( "", "" )
    Dim doc As NotesDocument
    Dim doc2 As NotesDocument
    Dim outline As NotesOutline
    Dim entry As NotesOutlineEntry
    Dim View As NotesView

    Dim agentLog As New NotesLog("Agent log")
    Call agentLog.OpenAgentLog
    Call agentLog.LogAction("Begin UpdateOutlineEntryWeb")

    Call db.Open( "Wtsc67oe", "PortalS390.nsf" )
    Set uidoc = workspace.CurrentDocument
    Set doc = uidoc.Document


    'Get the parameters data
    Call agentLog.LogAction("Before read parameters")
    FunctionalPage = doc.FunctionalPage(0)
    Options = doc.Options(0)
    TitleChoose = doc.TitleChoose(0)
    URLChoose = doc.URLChoose(0)
    FuncPage = doc.FuncPage
    OutlineMenu = doc.OutlineMenu
    Call agentLog.LogAction("After read parameters")
```

```
    'Get the corresponding Outline Menu
    NumElements = Ubound(FuncPage)
    Call agentLog.LogAction("Elements:= " + Cstr(NumElements))
    For i = 0 To NumElements
        If ( Cstr(FunctionalPage) = Cstr(FuncPage(i)) )Then
            Menu = OutlineMenu(i)
            Set outline = db.GetOutline(Cstr(Menu))
            Call agentLog.LogAction("Outline: " + Cstr(Menu))
            Exit For
        End If
    Next

    'Get the required entry
    Set entry = outline.GetFirst()

    If Not(entry Is Nothing) Then

        Do

            If ( Cstr(entry.Label) = Cstr(Options) ) Then

                'Set the label
                If ( Cstr(TitleChoose)  <> "") Then
                    entry.Label =  Cstr(TitleChoose)
                    Call agentLog.LogAction("Entry label changed")
                Else
                    Call agentLog.LogAction("Entry label not changed")
                End If

                'Set the new URL
                If  (Cstr(URLChoose) <> "") Then

                    Call agentLog.LogAction("To modify the URL")

                    If ( entry.SetURL(Cstr(URLChoose)) ) Then
                        Call agentLog.LogAction("URL modified")
                    Else
                        Call agentLog.LogAction("URL not assigned")
                    End If

                Else
                    Call agentLog.LogAction("URL not modified")
                End If

                ' Get the functional choose doc
                Set view = db.GetView("ViewFuncChoose")
                Set doc2 = view.GetFirstDocument

                Call agentLog.LogAction("looking for  the functional choose doc")
                If Not(doc2 Is Nothing) Then

                    Do
                        If (  (Cstr(doc2.FunctionalPage(0)) = Cstr(FunctionalPage)) And
(Cstr(doc2.ChooseName(0)) = Cstr(Options)) )Then

                            'Set the label
                            If ( Cstr(TitleChoose)  <> "") Then
                                doc2.ChooseName =  Cstr(TitleChoose)
                            End If

                            Call doc2.save(True, True)
```

```
                        Call agentLog.LogAction("Fuctional choose modified")
                        Exit Do

                    End If
                    Set doc2 = view.GetNextDocument(doc2)

                Loop While ( Not(doc2 Is Nothing) )

            End If

            Exit Do

        End If

        Set entry = outline.GetNext(entry)

    Loop While Not(entry Is Nothing)

End If

Call outline.save()

'Call db.UpdateFTIndex( False )
'Call workspace.ReloadWindow( )

Call agentLog.LogAction("End UpdateOutlineEntryWeb")
Call agentLog.Close

End Sub
```

A.5 The feedback application

Here some agents and Java code that we used in the feedback mechanism of the portal.

A.5.1 The WebSendMail agent

```
import lotus.domino.*;
import java.util.Vector;
public class JavaAgent extends AgentBase {
  public void NotesMain() {
    try {
        Session session = getSession();
        AgentContext agentContext = session.getAgentContext();
        Database db = agentContext.getCurrentDatabase();
         //get the feedback field
      Document doc = agentContext.getDocumentContext();
       String mytext = doc.getItemValueString("RichTextBody");
         //compose the memo
       Document memo = db.createDocument();
       memo.appendItemValue("Form", "Memo");
        memo.appendItemValue("Subject", "Feedback from Web client");
       memo.appendItemValue("Body", mytext);
         Vector v = new Vector();
        v.addElement(session.getUserName());
        v.addElement("armellecreuzet@itso.ibm.com");
       memo.setEncryptOnSend(true);
       memo.setSaveMessageOnSend(false);
```

```
        memo.setSignOnSend(true);
        memo.sign();
        memo.send(true, v);
      } catch(Exception e) {
      e.printStackTrace();
      }
   }
}
```

A.5.2 The NotesSendMail agent

```
Option Public

Sub Initialize
    Dim session As New NotesSession
    Dim workspace As New NotesUIWorkspace
    Dim uidoc As NotesUIDocument
    Dim doc As NotesDocument
    Dim memo As NotesDocument
    Dim subj As Variant
    Dim rtitem As Variant
    ' get the current doc
    Set uidoc = workspace.CurrentDocument
    Set doc = uidoc.Document
      'create a new document in the database
    Set memo = New NotesDocument(session.CurrentDatabase )
    ' set the new document's form so it'll be readable as a mail memo
    memo.Form = "Memo"
     ' set the new document's Subject
    memo.Subject = "Feedback from notes client"
     ' set the new document's Body
    'subj = doc.GetItemValue( "Body" )
    'memo.Body = subj(0)
    Set rtitem = doc.GetFirstItem( "RichTextBody" )
    If ( rtitem.Type = RICHTEXT ) Then
        memo.Body = rtitem.GetFormattedText ( False, 0 )
    End If

     ' send the new document to the recipient
    Call memo.Send( False, doc.GetItemValue( "SendTo" ))
End Sub
```

A.5.3 LiveNamesApplet Java code

```
import java.awt.*;
import java.applet.*;
import java.util.*;
import java.net.URL;
import com.lotus.sametime.core.comparch.STSession;
import com.lotus.sametime.core.comparch.DuplicateObjectException;
import com.lotus.sametime.awarenessui.list.AwarenessList;
import com.lotus.sametime.awarenessui.av.AVController;
import com.lotus.sametime.chatui.MeetingListener;
import com.lotus.sametime.chatui.MeetingInfo;
import com.lotus.sametime.chatui.ChatUI;
import com.lotus.sametime.community.*;
import com.lotus.sametime.lookup.*;
import com.lotus.sametime.core.types.STUser;
```

```
public class LiveNamesApplet extends Applet implements
LoginListener,ResolveListener,MeetingListener
{
/* Our session.*/
private STSession m_session;
/* The awareness list.*/
private AwarenessList m_awarenessList;
/* The entry point for the applet.*/
public void init()
{
try
{
// Generate a new session with a unique name
m_session = new STSession("LiveNamesApplet " + this);
// Call the session to load all available components
m_session.loadAllComponents();
m_session.start();
setLayout(new BorderLayout());
// Create the new list view
m_awarenessList = new AwarenessList(m_session, true);
add(m_awarenessList, BorderLayout.CENTER);
// The default right-click menu for the awareness list does not give audio & video as
options. We can change that by changing
// the controller to a different one.
AVController avController = new
AVController(m_awarenessList.getModel());
m_awarenessList.setController(avController);
// Get a reference to the ChatUI component in order to register a listener to the meeting
launched events. We use the session object
// which contains a reference to all the components that were loaded to get a reference to
the ChatUI component.
ChatUI chat = ((ChatUI)m_session.getCompApi(ChatUI.COMP_NAME));
chat.addMeetingListener(this);
// Login to the community
login();
}
catch(DuplicateObjectException e)
{
// This exception is thrown if an STSession with the same name has
// already been created.
e.printStackTrace();
}
}
/* Login to the community using the user name and password parameters from the html.*/
private void login()
{
// Get a reference to the community service. We use the session object
// which contains a reference to all the components that were loaded to
// get a reference to the community service.
CommunityService comm = (CommunityService)
m_session.getCompApi(CommunityService.COMP_NAME);
// Register a listener to the login/logout events.
comm.addLoginListener(this);
// Login to the community
//comm.loginByPassword("itsoebbers",
comm.loginByPassword(
//getCodeBase().getHost(),
getParameter("stserver"),
getParameter("loginName"),
getParameter("password"));
// Wait for the loggedin() event to start resolving and adding users to the list.
} // End login()
```

```
/* Logged in event. Add the users to the list.*/
public void loggedIn(LoginEvent event)
{
// Get a reference to the lookup service
LookupService lookup = (LookupService)
m_session.getCompApi(LookupService.COMP_NAME);
// Create a resolve object which will be used for resolving the user names into Sametime
user objects.
Resolver resolver = lookup.createResolver(false, // Return all matches.
false, // Non-exhaustive lookup.
true, // Return resolved users.
false); // Do not return resolved groups.
// Register a listener to receive the results of the resolve event.
resolver.addResolveListener(this);
// Get the list of users from the html
String users = getParameter("watchedNames");
// Parse the string in which user names are separated by commas.
StringTokenizer tokenizer = new StringTokenizer(users, ",");
String[] userNames = new String[tokenizer.countTokens()];
// Loop through all user names
int i = 0;
while(tokenizer.hasMoreTokens())
{
userNames[i++] = tokenizer.nextToken();
}
// Call the resolver to resolve the user names into Sametime user objects.
resolver.resolve(userNames);
} // End loggedIn

/* Users resolved event. Received from the resolver object as a response to a resolve
request. An event will be generated for each resolved user.*/
public void resolved(ResolveEvent event)
{
// A valid user was resolved, so add it to the list. We can safely cast to an array of
STUser since the resolve was constructed with users only flag.
m_awarenessList.addUser((STUser) event.getResolved());
} // End resolved()
/* Handle a resolve conflict event. Will be received when more than one match was found
for a specified user name.*/
public void resolveConflict(ResolveEvent event)
{
// Add all users to the list even if more than one match is found.
STUser[] users = (STUser[]) event.getResolvedList();
m_awarenessList.addUsers(users);
} // End resolveConflict()
/* Logged out event. Do nothing and rely on the default behavior, which will display a
dialog box.*/
public void loggedOut(LoginEvent event)
{
} // End loggedOut()
/* Resolve failed. No users are available to add to the list.*/
public void resolveFailed(ResolveEvent event)
{
} // End resolveFailed()
/* Open the meeting room client to participate in a meeting. This event is received as a
result of this user initiating a meeting or being invited to join a meeting by a different
user.*/
public void launchMeeting(MeetingInfo meetingInfo, URL url)
{
getAppletContext().showDocument(url,
meetingInfo.getDisplayName());
} // End launchMeeting()
/* An error has occurred during meeting creation.*/
```

```
public void meetingCreationFailed(MeetingInfo meetingInfo, int reason)
{
System.err.println("Failed to create the meeting: " + reason);
} // End meetingCreationFailed()
/* Applet destroyed. Stop and unload the session.*/
public void destroy()
{
m_session.stop();
m_session.unloadSession();
} // End destroy()
}
```

Appendix B. Using the additional material

This redbook also contains additional material on the Web. See the instructions below for using or downloading it.

B.1 Locating the additional material on the Internet

The material associated with this redbook is available in softcopy on the Internet from the IBM Redbooks Web server. Point your Web browser to:

ftp://www.redbooks.ibm.com/redbooks/SG246231

Alternatively, you can go to the IBM Redbooks Web site at:

ibm.com/redbooks

Select the **Additional materials** and open the directory that corresponds with the redbook form number.

B.2 Using the Web material

The additional Web material that accompanies this redbook includes the following:

File name	*Description*
sg246231.zip	Zipped files and directory as follows:

- PortalS390.nsf - Main Portal database and design
- domcfg.nsf- Contains the login form for the portal
- STPortal.nsf - Contains the java applet form STChat and login to Sametime
- Portalgl.nsf- The glossary that is used for Domino Global Workbench translation services
- The Applications directory contains the add-on application databases that were used to build the functional portal section.

B.2.1 System requirements for downloading the Web material

The following system configuration is recommended for downloading the additional Web material.

Hard disk space: 14 MB minimum

© Copyright IBM Corp. 2001

Operating System: OS/390 R6 or newer (tested with R10) along with Domino R5.04 or newer (tested with R5.06)

B.2.2 How to use the Web material

Create a subdirectory (folder) on your workstation and copy the contents of the Web material into this folder.

Appendix C. Special notices

This publication is intended to help application designers and Domino technical support engineers to create a portal using Domino in a S/390 environment. The information in this publication is not intended as the specification of any programming interfaces that are provided by Domino for S/390 R5. See the PUBLICATIONS section of the IBM Programming Announcement for Domino for S/390 R5 for more information about what publications are considered to be product documentation.

References in this publication to IBM products, programs or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM's product, program, or service may be used. Any functionally equivalent program that does not infringe any of IBM's intellectual property rights may be used instead of the IBM product, program or service.

Information in this book was developed in conjunction with use of the equipment specified, and is limited in application to those specific hardware and software products and levels.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact IBM Corporation, Dept. 600A, Mail Drop 1329, Somers, NY 10589 USA.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The information contained in this document has not been submitted to any formal IBM test and is distributed AS IS. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer's ability to evaluate and integrate them into the customer's operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers

© Copyright IBM Corp. 2001

attempting to adapt these techniques to their own environments do so at their own risk.

Any pointers in this publication to external Web sites are provided for convenience only and do not in any manner serve as an endorsement of these Web sites.

The following terms are trademarks of the International Business Machines Corporation in the United States and/or other countries:

e (logo)®
IBM ®
AIX
AS/400
AT
CICS
CT
Current
DB2
DB2 Connect
DFSMS/MVS
MQSeries
Netfinity
Open Class
OS/2
OS/390
OS/400
Parallel Sysplex
PR/SM
Processor Resource/Systems Manager
PR/SM
RACF
Redbooks
Redbooks Logo
Resource Link
RMF
RS/6000
S/390
SP
System/390
TXSeries
VisualAge
VM/ESA
WebSphere
XT
400
Lotus
Lotus Notes
Domino
Lotus QuickPlace
Notes
QuickPlace
Sametime

The following terms are trademarks of other companies:

Tivoli, Manage. Anything. Anywhere.,The Power To Manage., Anything. Anywhere.,TME, NetView, Cross-Site, Tivoli Ready, Tivoli Certified, Planet Tivoli, and Tivoli Enterprise are trademarks or registered trademarks of Tivoli Systems Inc., an IBM company, in the United States, other countries, or both. In Denmark, Tivoli is a trademark licensed from Kjøbenhavns Sommer - Tivoli A/S.

C-bus is a trademark of Corollary, Inc. in the United States and/or other countries.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and/or other

countries.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States and/or other countries.

PC Direct is a trademark of Ziff Communications Company in the United States and/or other countries and is used by IBM Corporation under license.

ActionMedia, LANDesk, MMX, Pentium and ProShare are trademarks of Intel Corporation in the United States and/or other countries.

UNIX is a registered trademark in the United States and other countries licensed exclusively through The Open Group.

SET, SET Secure Electronic Transaction, and the SET Logo are trademarks owned by SET Secure Electronic Transaction LLC.

Other company, product, and service names may be trademarks or service marks of others.

Appendix D. Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

D.1 IBM Redbooks

For information on ordering these publications, see “How to get IBM Redbooks” on page 271.

- *Building a Portal with Lotus Domino R5,* REDP0019 (redpaper)
- *Domino Certification Authority and SSL Certificates,* REDP0046 (redpaper)
- *Lotus Domino for Release 5: Enterprise Integration Using Lotus Connector Products*, SG24-5682
- *Lotus Domino for Release 5: Installation, Configuration and Administration*, SG24-2083
- *Enterprise Integration with Domino for S/390*, SG24-5150
- *Lotus Notes and Domino 5.0 Security Infrastructure Revealed*, SG24-5341
- *Connecting Domino to the Enterprise Using Java*, SG24-5425
- *Domino for S/390 and Web Server Integration*, SG24-5437
- *Lotus Domino R5.0 Enterprise Integration: Architecture and Products*, SG24-5593
- *Domino and WebSphere Together*, SG24-5955
- *Customizing QuickPlace*, SG24-6000

D.2 IBM Redbooks collections

Redbooks are also available on the following CD-ROMs. Click the CD-ROMs button at ibm.com/redbooks for information about all the CD-ROMs offered, updates and formats.

CD-ROM Title	Collection Kit Number
IBM System/390 Redbooks Collection	SK2T-2177
IBM Networking Redbooks Collection	SK2T-6022
IBM Transaction Processing and Data Management Redbooks Collection	SK2T-8038
IBM Lotus Redbooks Collection	SK2T-8039
Tivoli Redbooks Collection	SK2T-8044
IBM AS/400 Redbooks Collection	SK2T-2849

© Copyright IBM Corp. 2001

CD-ROM Title	Collection Kit Number
IBM Netfinity Hardware and Software Redbooks Collection	SK2T-8046
IBM RS/6000 Redbooks Collection	SK2T-8043
IBM Application Development Redbooks Collection	SK2T-8037
IBM Enterprise Storage and Systems Management Solutions	SK3T-3694

D.3 Other resources

These publications are also relevant as further information sources:

- *Administering the Domino System*, Lotus part numbers CT7VHNA and CT7VINA, `http://www.lotus.com/yellowbooks`
- *Managing Domino Databases,* Lotus part number CT73MNA, h`ttp://www.lotus.com/yellowbooks` or `http://notes.net/notesua.nsf`
- Domino Global Workbench: Guidelines for Developers and Translators
- Domino Global Workbench Help (dgwhelp.hlp, comes with product)
- Java-Sametime: What's New in the Sametime 2.0 Java Toolkit? (whitepaper)
- Sametime 2.0 Administration Help (sthelpad.nsf, comes with product)
- Sametime 2.0 Installation guide (with product)
- Domino 5 Administration Help (with product)
- Domino 5 Designer Help (with product)

D.4 Referenced Web sites

These Web sites are also relevant as further information sources:

- `http://www.lotus.com/home.nsf/welcome/globaldesgnr`
- `http://www.dkms.com/EIPDEF.html` ("Enterprise Information Portal Definition Is a Political Process")
- `http://www.notes.net/today.nsf` ("SSL: It's not just for commerce anymore")
- `http://www.lotus.com/home.nsf/welcome/eizone` (Lotus enterprise integration)
- `http://www.notes.net/today.nsf` (Domino NNTP server)
- `http://notes.net/notesua.nsf/0b345eb9d127270b8525665d006bc355/05af8a79099694b685256a34005314d9?OpenDocument` (Domino for S/390 documentation)
- `http://www.ibm.com/redbooks`
- `http://www.lotus.com/sametime`

How to get IBM Redbooks

This section explains how both customers and IBM employees can find out about IBM Redbooks, redpieces, and CD-ROMs. A form for ordering books and CD-ROMs by fax or e-mail is also provided.

- **Redbooks Web Site** ibm.com/redbooks

  Search for, view, download, or order hardcopy/CD-ROM Redbooks from the Redbooks Web site. Also read redpieces and download additional materials (code samples or diskette/CD-ROM images) from this Redbooks site.

  Redpieces are Redbooks in progress; not all Redbooks become redpieces and sometimes just a few chapters will be published this way. The intent is to get the information out much quicker than the formal publishing process allows.

- **E-mail Orders**

  Send orders by e-mail including information from the IBM Redbooks fax order form to:

	e-mail address
In United States or Canada	pubscan@us.ibm.com
Outside North America	Contact information is in the “How to Order” section at this site: http://www.elink.ibmlink.ibm.com/pbl/pbl

- **Telephone Orders**

United States (toll free)	1-800-879-2755
Canada (toll free)	1-800-IBM-4YOU
Outside North America	Country coordinator phone number is in the “How to Order” section at this site: http://www.elink.ibmlink.ibm.com/pbl/pbl

- **Fax Orders**

United States (toll free)	1-800-445-9269
Canada	1-403-267-4455
Outside North America	Fax phone number is in the “How to Order” section at this site: http://www.elink.ibmlink.ibm.com/pbl/pbl

This information was current at the time of publication, but is continually subject to change. The latest information may be found at the Redbooks Web site.

IBM Intranet for Employees

IBM employees may register for information on workshops, residencies, and Redbooks by accessing the IBM Intranet Web site at http://w3.itso.ibm.com/ and clicking the ITSO Mailing List button. Look in the Materials repository for workshops, presentations, papers, and Web pages developed and written by the ITSO technical professionals; click the Additional Materials button. Employees may access MyNews at http://w3.ibm.com/ for redbook, residency, and workshop announcements.

© Copyright IBM Corp. 2001

IBM Redbooks fax order form

Please send me the following:

Title	Order Number	Quantity

First name	Last name	
Company		
Address		
City	Postal code	Country
Telephone number	Telefax number	VAT number

☐ Invoice to customer number _____

☐ Credit card number _____

Credit card expiration date	Card issued to	Signature

We accept American Express, Diners, Eurocard, Master Card, and Visa. Payment by credit card not available in all countries. Signature mandatory for credit card payment.

Glossary

ACL Access control list. A list of database users (individual users, Lotus Domino servers, and groups of users and/or servers) created and updated by the database manager. The ACL specifies which users can access the database and what tasks they can perform.

Address space An OS/390 construct used to represent a batch job, TSO/E user, started task, UNIX Systems Services user, or forked USS process. An address space control block (ASCB) points to many other control blocks used to manage the virtual storage, I/O resources and programs used by the work unit. A USS process creates an address space.

ADSM ADSTAR Distributed Storage Manager. IBM product for distributed file backup, restore, archive and more. The products have been rebranded to Tivoli Data Manager (TDM).

APAR Authorized Program Analysis Report. IBM record of a software problem.

Canonical Format Format for storing hierarchical names that displays the hierarchical attribute of each component of the name. For example, the canonical format for the name Reuben Smith/ Ottawa/Acme/CA is: CN=Reuben Smith/OU=Ottawa/O=Acme/C=CA where: CN is the common name, OU is the organizational unit, O is the organization, and C is the country code.

CMG Computer Measurement Group. They hold conferences on technology management and performance evaluation.

DADSM Direct Access Device Space Management. Routines of the OS/390 operating system that manage disk storage.

DAE Dump Analysis and Elimination.

DECS Domino Enterprise Connection Services. A forms-based interface that allows integration to external data from Domino applications.

DFSMS Data Facility Storage Management Subsystem. A software suite that automatically manages data from creation to expiration. DFSMS/MVS provides allocation control for availability and performance, backup/recovery and disaster recovery services, space management, tape management, and reporting and simulation for performance and configuration tuning.

DNN Domino Named Network

DNS Domain Name System. A function to associate names and addresses on Internet domain servers.

Domain A Domino domain is a collection of Domino servers and users that share a common Domino directory. The primary function is mail routing. Users' domains are determined by the location of their server-based mail files.

GAN Global area network.

GRS Global Resource Serialization. An OS/390 base component that manages serialization.

HTTP Hypertext Transmission Protocol. An Internet protocol used to transfer files from one computer to another.

ICM Internet Cluster Manager. Domino component for failover and load balancing of HTTP clustering.

IMAP Internet Message Access Protocol. A mail protocol that allows clients running it to retrieve mail from a host mail server also running the protocol. IMAP is similar to POP3 but has additional features. For example, it supports three modes of mailbox access. You can enable IMAP on a Domino server.

I/O Input/output.

IPCS Interactive Problem Control System. An OS/390 tool for examining operating system and subsystem dumps. The OS/390 flavor of IPCS is written in all caps.

ipcs Inter-Process Communication System. A UNIX programming model to coordinate and serialize separate threads. The UNIX ipcs is usually written in lower case characters.

© Copyright IBM Corp. 2001

IPL Initial program load. The initialization procedure that causes an operating system to start operation.

IRIS A wholly owned division of Lotus. They developed the Domino/Notes product.

ITSO International Technical Support Organization.

JES2 Job Entry Subsystem/2. A component of OS/390 that manages the initiation and termination of batch jobs, TSO users and started tasks as well as printed output.

LAN Local area network.

LDAP Lightweight Directory Access Protocol. A set of protocols for accessing information directories. LDAP is based on the X.500 protocol, but supports TCP/IP, which is necessary for Internet access. Because it's a simpler version of X.500, LDAP is sometimes called X.500-lite. You can enable LDAP on a Domino server to allow LDAP clients to access information in Domino directory, for example, e-mail addresses.

Logical partition In LPAR mode, a subset of the processor unit resources that is defined to support the operation of a system control program (SCP).

Logically partitioned mode (LPAR) A mode that allows the operator to allocate hardware resources of the processor unit among several logical partitions.

Lotus A wholly-owned subsidiary of IBM, and the owner of the Domino/Notes product.

LotusScript A version of BASIC that offers not only standard capabilities of structured programming languages, but a powerful set of language extensions that enable object-oriented development within and across products. Its interface to Notes is through predefined object classes.

LPAR Logical Partition. System/390 capability to divide up the processors, memory and channels so that several operating system images can run independently on the same machine.

LS:DO The ODBCConnection, ODBCQuery, and ODBCResultSet classes, collectively called the LotusScript Data Object (LS:DO), provide properties and methods for accessing and updating tables in external databases through the ODBC (Open Database Connectivity) Version 2.0 standard.

MIB Management Information Base. Defines the scope of what can be managed by SNMP.

MIME Multipurpose Internet Mail Extensions. Software that allows you to attach non-text files to Internet mail messages. Non-text files include graphics, spreadsheets, formatted word-processor documents, and sound files.

MTA Mail transfer agent. A message transfer agent, also called a gateway, is a program that translates messages between mail formats.

NAB Notes Name and Address Book. Now called the Domino directory or Personal Address Book.

NNN Notes Named Network. Now DNN or Domino Named Network.

NNS Notes Name Service. Domino service for converting Fully Qualified Domain Names or Host names to a TCP/IP address.

NNSC Notes Name Service Cache. Hidden fields in the user's personal NAB Location document that is used when accessing the Domino server in question or in the server document of the server that initiated the connection to the other Domino server.

NNTP Network News Transfer Protocol. Protocol that supports reading newsgroups, posting new articles, and transferring articles between news servers. When enabled on a Domino server, allows NNTP clients to access newsgroups on the server and allows the Domino server to exchange news with other NNTP servers.

NRPC or **Notes RPC** Notes remote procedure call. This is the architectural layer of Notes used for all Notes-to-Notes communication. You can set up either the HTTP or the SOCKS proxy to work with RPC.

NSF The file extension for a Notes database file (.NSF). A database file contains the data for an application. Its structure is composed of forms,

fields, folders, views, and other presentation features, such as a navigator and a database icon.

NTF The file extension for a Notes template file (.NTF). A template file contains the structure for the database--that is, forms, folders, and views--but does not contain documents. Domino Designer comes with a collection of templates that you can use to create system and application databases.

NTI Notes Transport Interface. The component of Domino server that manages communications.

ODBC Open Database Connectivity. A standard developed by Microsoft for accessing external data. ODBC has four components: the ODBC-enabled application, the ODBC Driver Manager, ODBC drivers, and data sources. Lotus Notes is an ODBC-enabled application.

ODS On Disk Structure. The format in which a Notes database is physically stored on disk. Also called a database format. In Release 5, the ODS version of a database is listed on the Info tab of the Database Properties box.

POP3 Post Office Protocol (level 3). A mail protocol that allows clients running it to retrieve mail from a host mail server also running the protocol. You can enable POP3 on a Domino server.

PTF Program Temporary Fix. A temporary solution or by-pass for a problem diagnosed by IBM as resulting from an error in a current unaltered release of the program.

QMR Quarterly Maintenance Release. Lotus software upgrade package, distributed quarterly, to give customers the latest release level of Domino/Notes code. The entire package must be installed, as it is a complete code replacement.

QMU Quarterly Maintenance Upgrade. Lotus software upgrade package distributed between QMRs and designated by a letter, for example 5.0.4a. It is for critical or widespread fixes to Domino code.

RMF Resource Measurement Facility. Monitors the activity of the OS/390 system. It is useful for capacity planning, performance tuning, and problem diagnosis. RMF gives information about system resources, workload groups, DASD controllers, and DASD volumes.

SIMS Support Information Management System. Lotus internal databases for tracking problems

SMF System Management Facilities. OS/390 facility for recording performance and capacity data.

S/MIME Secure/MIME. A secure version of the MIME protocol that allows users to send encrypted and electronically signed mail messages, even if users have different mail programs.

SDSF Spool Display Search Facility. OS/390 tool for JES2 systems that allows a TSO user to view and route output from the JES2 spool and to view the SYSLOG and issue commands to OS/390 if authorized.

SLA Service level agreement.

SMTP Simple Mail Transfer Protocol. The Internet standard host-to-host mail transport protocol. It traditionally operates over TCP/IP using port 25. SMTP does not provide a mailbox facility, or any special features beyond basic mail transport.

SNMP Simple Network Management Protocol. A TCP/IP protocol to enable managing remote hosts.

SPR Software Problem Record. Lotus software problem record tracking record.

SSL Secure Sockets Layer. Security protocol for the Internet and intranets that provides communications privacy and authentication for Domino server tasks that operate over TCP.

TCB Task Control Block. An OS/390 construct that represents a dispatchable unit of work. One address space can create multiple tasks. A USS thread creates a TCB in OS/390.

TCP/IP Transmission Control Protocol/Internet Protocol. Network protocols that define the Internet. Originally designed for UNIX, TCP/IP software is now available for every major computer operating system.

TDP Tivoli Data Protection.

terabyte A trillion characters of data.

TSO/E Time Sharing Option/Extended. The principal interactive interface to OS/390.

UNIX An operating system developed at Bell Laboratories (trademark of UNIX System Laboratories, licensed exclusively by X/Open Company, Ltd.).

URL Uniform Resource Locator

VIPA Virtual Internet Protocol (IP) addressing. A virtual TCP/IP address that can direct network traffic to more than one physical adapter.

VTOC Volume Table of Contents. A list on each disk volume of the files and space it contains.

WAN Wide area network.

Index

Numerics

24-hour access with S/390 22

A

access
  24 hours 22
  rate 22
access control 77
  best practices 100
  database 98
  list (ACL) 98
    defined 55
access control list (ACL) 127
  content 33
  user roles 35
AddOutlineEntryWeb 145
AddOutlineEntryWeb agent 145
addressing, S/390 storage 16
administration 26
  cost 16
  effort 16, 19
    larger server 16
  of portal 43
  pages 134
application
  collaborative 42
  invoice 42
authentication 77
  Domino server 77
  HTTP server 78
  Internet/intranet clients 79
  session-based 82
  SSL 96
automation tools 21
availability 19, 21
  S/390 21

B

backup 23
  speed 23
    on S/390 23
benefits
  of a portal 3, 9
  of S/390 13
    See also S/390 benefits
bottlenecks, removing 16
building a portal 9
business
  cards 119
  contacts 108

C

cache, FRCA 48
calendaring and scheduling 107, 118
capability
  connected users 15
  powerful S/390 I/O 23
cataloguer task 126
certification
  third-party 91
challenge/response 77
chat function 112
CICS 26, 190, 193
classes
  description of connector 176
coding for Notes client and Web browser 48
collaborative applications 42
concepts 172
  of a portal 3
connected users, S/390 15
connector
  classes 176
  products 159
    declarative 159, 175
    programmatic 175
consolidation
  hardware 20
contacts 108, 119
content mapping with access control 34
cookie 82
CORBA 194
corporate portal 5
costs
  hardware 21
  low 21
  software 21
  support staff 21
current
  data, accessed on large server 17
  events 110, 121
customer service 22

© Copyright IBM Corp. 2001

D

data 21
  access 21
  backup 23
    speed 23
  integrity 21
  management on S/390 23
  security 21
database
  access control 98
  administrator 56
  discussion 129
  of the portal 33
  translation process 61
DB2 26
declarative access 175
DECS
  administrator database 167
  installation 163
design
  changes 72
  of the administration section 43
  of the enterprise portal 40
  of the functional portal 42
  of the personal portal 41
  of the portal kernel 31
development using one server 20
discussion database 129
  NNTP 211
disk space 17
  larger server 17
document library application 130
domain
  catalog 122
  indexer 122
  search 122
Domino 26
  administration 14
  Administrator client 123
  benefits on S/390
  beside enterprise applications 19
  close to enterprise data 19
  Designer 45, 194
  domain search 122
  for IBM HTTP Server 48
  Global Workbench (DGW) 60
  integration with OS/390 Web server 48
  powerful functionality 13
  support plan 26
  value of large server 15
Domino Enterprise Connection Services (DECS) 159
  concepts 163

E

electronic filing cabinet 130
e-mail 106, 117
enterprise
  content 30
  news 128
  portal design 40
  portal page 122
enterprise integration
  connector products 159
  DB2 using DECS 162
  DB2 using LEI 172
  tools 161
error detection 19
expanded processor storage 16
experience, none 23
extranet content 30

F

features of a portal 6
feedback 151
field mapping 168
finding the portal 50
flexibility of S/390 22
floating point 25
formula language, when to use 46
frameset 37
FRCA cache 48
functional portal
  design 42
  pages 133
fuzzy search 124

G

GenerateWebPersonalPage 112
Global Workbench 60
globalization 59
glossary
  database 63
  language view 71
groups 99
growth 22

S/390 22
guarantee
integrity of MVS 21

H

handshake 92
hardware
consolidation 20
S/390 19
headlines 110, 121
horizontal portal 4
hotspot resource link 108

I

I/O capability 23
IBM WebSphere Translation Server 60
identifying a workgroup 26
IEEE floating point hardware 25
IMS 26
integrity 21
guarantee in MVS 21
one server 21
interest profile of NNTP 215
Internet
content 31
security 96
inter-server traffic 16
intranet 21
invoice application 42
IPL
OS/390 19
isolation 20

J

Java 194
applets 45
when to use 46
JavaScript 45, 194
when to use 46
javascript 88
JDBC 194
job scheduler 16

K

kernel
code, OS/390 19
portal 31
key fields 171
knowledge management elements 9

L

large Domino server
value 15
LCConnection 177
class description 177
LCFieldlist 177
class description 177
LCLSX 175
agent 180
programming connection 42
LCSession 177
class description 177
LDAP server 234
library application 130
list server 110, 121
login interface 44
Lotus
connectors
directory services 160
ERP 160
OLTP 160
other services 160
products 159
RDBMS 160
Lotus Connector LotusScript Extension (LCLSX) 160
described 176
Lotus Enterprise Integrator (LEI) 160, 172
Lotus Notes deployment 26
Lotus Translation Component 60
LotusScript, when to use 46
low costs with one server 21
low risk, S/390 23

M

mail 106, 117
routing with larger server 17
megaportal 4
menu generator
components 141
MQGetMessageOptions 189
MQMessage 189
MQProcess 189
MQPutMessageOptions 189
MQQueue 189

MQQueueManager 189
MQSeries 186
MQSeries LotusScript Extension (MQLSX) 186
MQSeries Message Queueing Interface (MQI) 186
MQSession class 189
multi-lingual
  database generation 62
  portal 59
    tools 60
multiple operating systems on one server 20
multivalue field 166
MVS
  integrity guarantee 21

N

navigation 134
network
  connections 19
  traffic 44
    larger server 17
network news transfer protocol (NNTP) 197
  newsfeeds 201
  security with Domino 200
  using Domino 199
news
  enterprise 128
  viewing 212
news subscription 110, 121
newsfeeds 197
  managing 201
newsgroups 197
  serving 208
no experience 23
Notes client 106
  compatible code with Web browser 48

O

one server
  multiple operating systems 20
online sales 59
opening the portal 50
  from a browser 51
  from the Notes client 50
OS/390
  kernel code 19
  one server 20
  Security Services (RACF) 21
    See also RACF
    See also SAF
  support team 25

P

P/390
  card 22
page types
  administration 134
  enterprise 122
  functional 133
  personal 105
parallel sysplex 22
parts ordering application 190
PC server 21
  availability 21
  reliability 21
peak loads 19
performance 19, 21
personal
  content 29
  mail 117
  portal
    design 41
    pages 105
personalization 55
  of portal desktop 59
pervasive devices 7
portal
  administration pages 134
  administrator 138
  benefits 3, 9
  building a 9
  chat 226
  concepts 3
  configuration 134
  connectors
    declarative 159
    programmatic 175
  content
    enterprise 30
    extranet 30
    Internet 31
    personal 29
  corporate, user tools 5
  customer application 165
  database 33
    administrator 56
  DECS 164

design 31
enterprise page 122
features 6
feedback 151
finding 50
functional page 133
horizontal 4
invoice application 178
kernel 31
main page 36
menu structure 37
multi-lingual 59
news 128
newsgroups 212
parts ordering application 190
personal pages 105
frameset structure 112
Notes client 106
Web browser client 112
personalization 55
desktop 59
user roles 55
QuickPlace 227
Sametime 217
security 44, 77
structure 32
user interface 47
vertical 4
postings, receiving 110, 121
powerful I/O capability
S/390 23
PR/SM 20
See also Processor Resource/Systems Manager
processor
capacity 18, 19
larger server 18
peak usage 18
Processor Resource/Systems Manager 20
programmatic connectors 175
concepts 175
PTF 25
checker 25
public address book
distribution 16
larger server 16

Q

QuickPlace 227

R

RACF 21
See also SAF
See also System Authorization Facility
See OS/390 Security Services 21
realms 89
rebooting
OS/390 19
reference database 130
registering users 56
re-IPLing
OS/390 19
reliability 21
S/390 21
RemoveOutlineEntryWeb agent 147
risk
low for S/390 23
S/390 23
failure 23
roles 99
assigning 56
RSA encryption 94

S

S/390 benefits 13
access to current data 17
administration 16
large Domino server 16
low cost 21
low risk 23
mail routing 17
multiple applications on one server 18
multiple operating systems 20
reliability and availability 21
scalability 22
security and integrity 21
side-by-side applications 19
single server 15
speed 23
S/390 server
hardware 19
supports large workloads 16
SAF 21
See also System Authorization Facility
Sametime 217

Java applets 221
setup 218
scalability 13
search 122
fuzzy 124
stemming 124
Secure Sockets Layer (SSL) 91, 199
security 19, 21, 44, 77
internet 96
one server 21
services 92
mutual authentication 92
session integrity 92
server
multiple applications 18
multiple operating systems 20
splitting users 22
upgrading 22
service strategy for Domino 26
session based authentication 83
single server, benefits of 15
SMP 16
See also symmetric multiprocessors
SMP/E 25
speed 23
SQL 179
SSL
authentication 96
benefits 94
handshake 92
stemming 124
storage addressing
S/390 16
structure of the portal 32
subscriptions 110, 121
support
connected users 15
plan 26
staff, OS/390 23
support team for OS/390 25
symmetric multiprocessors (SMPs) 16
system
operation 21
resources 20
System Authorization Facility (SAF) 21

T

tagged database 64
task, inter-server 16
teamroom 42, 133
teamware 228
third-party certification 91
to do list 109, 120
tools, automation 21
TPF, one server 20
Translation Component 60
translation of database 61
Translation Server 60

U

UNIX reliability and availability 21
UNIX System Services 14
UpdateOutlineEntryWeb agent 149
upgrading a server 22
USENET 197
user
connected 15
feedback 151
interface 47
largest number of 15
roles 55

V

vertical portal 4
VM, one server 20
VSAM file 192

W

Web browser
client 112
compatible code with Notes client 48
Web server integration with Domino 48
Web site 59
translation 60
Webmaster 152
WebSphere Translation Server 60
workgroup 29, 129
workload
isolation 19
one server 20
manager 19
World Wide Web (WWW) 22

X

X.509 certificates 93

IBM Redbooks review

Your feedback is valued by the Redbook authors. In particular we are interested in situations where a Redbook "made the difference" in a task or problem you encountered. Using one of the following methods, **please review the Redbook, addressing value, subject matter, structure, depth and quality as appropriate.**

- Use the online **Contact us** review redbook form found at **ibm.com**/redbooks
- Fax this form to: USA International Access Code + 1 845 432 8264
- Send your comments in an Internet note to redbook@us.ibm.com

Document Number **Redbook Title**	SG24-6231-00 Build a Portal with Domino: A S/390 Example
Review	
What other subjects would you like to see IBM Redbooks address?	
Please rate your overall satisfaction:	O Very Good O Good O Average O Poor
Please identify yourself as belonging to one of the following groups:	O Customer O Business Partner O Solution Developer O IBM, Lotus or Tivoli Employee O None of the above
Your email address: The data you provide here may be used to provide you with information from IBM or our business partners about our products, services or activities.	O Please do not use the information collected here for future marketing or promotional contacts or other communications beyond the scope of this transaction.
Questions about IBM's privacy policy?	The following link explains how we protect your personal information. **ibm.com**/privacy/yourprivacy/

© Copyright IBM Corp. 2001

Build a Portal with Domino: A S/390 Example

(0.5” spine)
0.475”<->0.875”
250 <-> 459 pages

Build a Portal with Domino: A S/390 Example

The concepts, features, and benefits of a portal as a knowledge tool

Building your own portal architecture with levels of content

Integrating applications into the portal

In today's fast-paced knowledge management environment, a great deal of importance is placed on providing professional workers with information that is tailored to their job responsibilities. This IBM Redbook gives a step-by-step description of how to build a portal application using Lotus Domino. While our sample was built in an OS/390 environment using UNIX System Services, the concepts and code samples apply equally well to any platform that supports a Domino server.

The audience for this redbook includes Domino developers, architects, and technical support engineers on all platforms, as well as consultants and marketing support professionals. A knowledge of Domino, programming, and operating systems is assumed.

INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION

BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

For more information:
ibm.com/redbooks

SG24-6231-00 ISBN 0738422401